C# da Diziler

Diziler için aynı tipteki verilerin tutulduğu bir koleksiyon diyebiliriz. Örneğin integer verinin bir yığın şeklinde tutulması için dizileri kullanırız. C# da diziler referans tipinde değişkenlerdendir. C# da tanımlanan tüm diziler System.Array sınıfından türemiş bir nesnedir. C# da diziler aşağıdaki gibi tanımlanır.

```
<veri tipi>[] <değişken ismi> = new <veri tipi>[<dizinin boyutu>];
```

10 adet integer veri tutan bir dizinin tanım ise

```
int[] integerDizi = new int[10];
```

Bir dizinin boyutları sabittir ve kullanılmadan önce belirlenmelidir. Dizi boyutunu belirlemek için başka bir değişkende kullanabilirsiniz.

```
int boyut = 10;
int[] integerDizi = new int[boyut];
```

Diziyi tanımlama ve başlangıç değerlerini atama işlemini ayrı satırlardada yapabilirsiniz.

```
int[] integerDizi;
integerDizi = new int[10];
```

Ayrıca dizileri tanımlarken, dizi içine atmak istediğiniz verileride belirterek dizi tanımlayabilirsiniz. Bunun için kullanacağınız veri tipine uygun olacak şekilde, süslü parantez içinde her biri virgülle ayrılmış dizi elemanlarını yazmanız yeterli.

```
int[] integerDizi = {1,2,3,4,5,6,7,8,9};
```

Yukarıda 10 adet elemanı olan ve değerleri de verilmiş, integer tipinde verileri tutan bir dizi tanımladık. Eğer dizimiz string değerler tutacak olsaydı, süslü parantez içine yazdığımız elemanların her birini çift tırnaklar arasına almamız gerekirdi.

```
string[] strDizi = { "Sabri", "Metin", "Osman", "Ali" };
```

Dizi Elemanlara Ulaşmak

Dizi elemanlarına ulaşmak için [indeks] indeks operatörünü kullanırız. Dikkat edilmesi gereken nokta C# da dizilerin elemanları 0. İndeksten başlar, yani eğer 5 elemanlı bir dizimiz varsa bu dizinin birinci elemanı 0. indekste son elemanı ise 4. indekstedir. 5 elemanlı bir dizinin 3. elemanına aşağıdaki gibi erişiriz.

```
int[] integerDizi = { 1, 2, 3, 4, 5};
int ucuncuEleman = integerDizi[2];
```

Bir dizi içindeki elemanlara ulaşmak için basit bir örnek:

```
int[] integerDizi = { 4, 8, 23, 64, 35 };
for (int i = 0; i < 5; i++)
{
Console.WriteLine("Dizinin {0}. Elemanının Değeri = {1} ", i,
integerDizi[i]);
}
```

Yukarıdaki kodun çalışması sonucu oluşan ekran görüntüsü:

Yukarıdaki örnekte bir dizi oluşturduk ve bu dizinin eleman sayısı kadar bir for döngüsü kurduk. Döngü içinde diziye ait elemanların değerlerine tek tek ulaştık.

Peki kullandığımız dizinin eleman sayısını bilmeseydik nasıl döngüye girebilirdik. Bütün diziler System.Array sınıfından türemiş nesnelerdir demiştik. System.Array sınıfının Length diye bir özelliği vardır, doğal olarakda tüm dizilerinde bir **Length** özelliği olur. Dizimizin eleman sayısını **dizininAdi.Length** diyerek alabiliriz. Yukarıdaki örneği bu yöntemle yeniden yazacak olursak.

```
int[] integerDizi = { 4, 8, 23, 64, 35 };
for (int i = 0; i < integerDizi.Length; i++)
{
Console.WriteLine("Dizinin {0}. Elemanının Değeri = {1} ", i,
integerDizi[i]);
}
```

Yukarıdaki kodu çalıştırdığınızda bir önceki örneğimizle aynı sonucu üretir.

Dizilerle ilgili dersimize devam etmeden önce diziler için kullandığımız foreach döngüsüne bakalım.

foreach Döngüsü

Diziler ve koleksiyonların elemanlarına erişmek için basit ve kullanışlı bir döngü çeşidi. Döngüler dersinde bahsetmedik çünkü dizileri anlatmadan foreach döngüsünü anlamaya çalışmak boşuna olacaktı. foreach döngüsünün yapısı aşağıdaki gibidir.

```
foreach(<dizi içindeki elemaların veri tipi> <değişken ismi> in
<diziveyaKoleksiyonadi>)
{
<Kod blogumu>
}
```

Yukarıda for döngüsü ile yaptığımız örneğimiz foreach döngüsü ile yapalım.

```
foreach (int diziElemanı in integerDizi)
{
Console.WriteLine("Elemanın Değeri = {0} ", diziElemanı);
}
```

Çok daha basit ve kullanışlı olduğunu görebilirsiniz.

- Foreach döngüsü kullanırken ulaştığımız dizi elemanları readonly dir. Yani yukarıdaki örnek üzerinde anlatacak olursak foreach döngüsü içinde diziElemanı adlı integer değişkenin içerdiği değeri değiştiremezsiniz. Eğer değiştirmeye kalkarsanız `Cannot assign to 'diziElemanı' because it is a 'foreach iteration variable'` hatası alırsınız. Eğer yazdığınız uygulamada dizi elemanlarını döngü içinde değiştirmek istiyorsanız for döngüsü kullanmanızı tavsiye ederim.
- String tipindeki bir değişkende char tipindeki verilerin birleşmesinden oluşmuş bir dizidir. Öyleyse foreach döngüsü kullanarak aşağıdaki gibi bir stringin elemanlarını erişebilirsiniz.

```
string metin = "Yazılım Mutfağı";
foreach (char harf in metin)
{
Console.WriteLine("Harf = {0} ", harf);
}
```

Çok Boyutlu Diziler (Multidimensional Array)

Çok boyutlu bir dizi 'dizinin dizisi' şeklinde tabir edilebilir. Her elemanı başka bir dizi olan diziler çok boyutlu bir dizidir. Çok boyutlu bir dizi veritabanında her satırında birden fazla kolon bulunan bir tabloya benzetilebilir. Eğer her satırda bulunan her kolon kendi içinde başka bir dizi barındırmıyorsa bu tür dizilere 2 boyutlu diğer bir değişle çok boyutlu diziler denir. Eğer her kolonda kendi içinde başka bir dizi barındırıyorsa bu tür dizilere n boyutlu diziler denir. Burda n ifadesi dizinin derinliğini ifade eder.

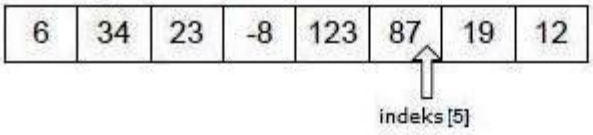
Çok boyutlu diziler iki çeşittir;

- Düzenli Diziler: Her satırı aynı sayıda kolon barındıran diziler.
- Düzensiz Diziler (Jagged Array) : Her satırı farklı sayıda kolon barındıran diziler.

Çok boyutlu dizileri şekillerle anlatmaya çalışalım.

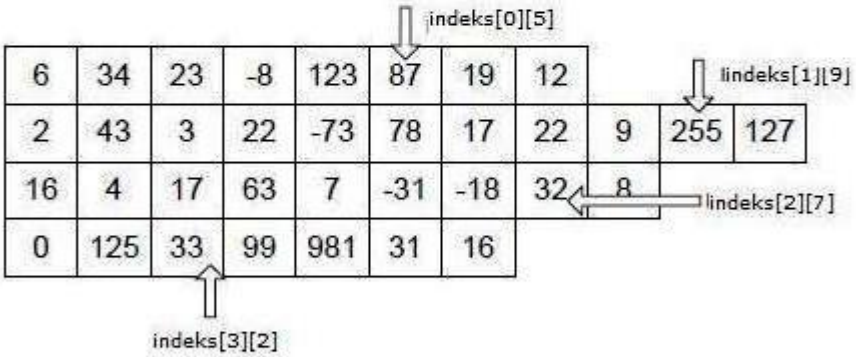
Not: Dizilerin ilk elemanının indeks numarasının 0 olduğunu unutmayın.

Tek Boyutlu Bir Dizi

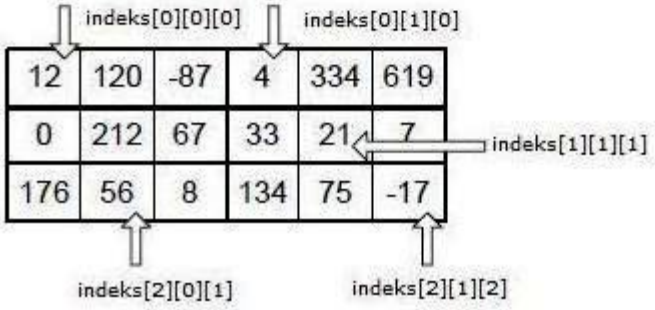


2 Boyutlu(Düzenli) Bir Dizi

2 Boyutlu(Düzensiz) Bir Dizi



3 Boyutlu Bir Dizi



Dizilerin Tanımlanması ve Değer Atamaları

Tek boyutlu bir dizinin nasıl tanımlandığını ve nasıl değer atayacağımızı yukarda anlatmıştık. Şimdi Tek boyutlu dizi dahil tüm dizi tiplerini nasıl tanımlayacağımıza bakalım.

Tek Boyutlu Diziler:

```
//Dizinin Oluşturulması
```

```
int[] intDizi = new int[5];
//Dizi elemanlarına değer ataması
intDizi[0] = 36; //dizinin ilk elemanına 36 değeri atandı
intDizi[2] = 26; //dizinin ikinci elemanına 26 değeri atandı
intDizi[4] = 32; //dizinin son elemanına 32 değeri atandı
```

Çok Boyutlu Diziler:

```
//Dizinin Oluşturulması
int[,] intCokBDizi = new int[2, 3];
//Dizi elemanlarına değer ataması
intCokBDizi[0, 0] = 23;
intCokBDizi[0, 1] = 25;
intCokBDizi[0, 2] = 33;
intCokBDizi[1, 0] = 41;
intCokBDizi[1, 1] = 29;
intCokBDizi[1, 2] = 93;
```

Çok Boyutlu Dizilerin Elemanlarına Döngü İle Ulaşmak

for döngüsü:

```
//Dizi elemanlarını for döngüsü ile bulmak.
for (int row = 0; row < intCokBDizi.GetLength(0); row++)
{
for (int col = 0; col < intCokBDizi.GetLength(1); col++)
{
Console.WriteLine("Dizinin {0},{1} indeksindeki değeri :
{2}",row,col,intCokBDizi[row,col]);
}
}
```

Ekran Çıktısı:

```
Dizinin 0,0 indeksindeki değeri : 23
Dizinin 0,1 indeksindeki değeri : 25
Dizinin 0,2 indeksindeki değeri : 33
Dizinin 1,0 indeksindeki değeri : 41
Dizinin 1,1 indeksindeki değeri : 29
Dizinin 1,2 indeksindeki değeri : 93
```

GetLength() fonksiyonu dizinin belirtilen boyutundaki eleman sayısını verir. Yukarıdaki örnekte görüldüğü gibi parametre olarak 0 veya 1 veriyoruz. 0 dizinin birinci boyutundaki eleman sayısı yani satır sayısı, 1 verdiğimizde ise dizinin 2.boyutu yani kolon sayısını verir.

Dizilerin Length özelliği dizi içinde bulunan toplam eleman sayısını verir. Örneğin yukarıdaki örnekte tanımladığımız dizi için `intCokBDizi.Length` sonuç olarak 6 döndürür.

2 boyutlu Düzensiz dizilerde foreach döngüsü kullanamayız. foreach döngüsünü sadece tüm elemanlara erişmek için kullanabiliriz. Eğer her boyuttaki elemana farklı erişmek istiyorsak for döngüsü kullanmamız gerekiyor. foreach döngüsü ile düzensiz 2 boyutlu dizilerin elemanlarına erişebiliriz. Önce düzensiz dizilerin nasıl tanımlandığına bakalım sonrada foreach döngüsü ile elemanlarına nasıl erişeceğimize göz atalım.

Düzensiz Çok Boyutlu Diziler:

```
//Dizinin Oluşturulması
int[][] intCokBDizi = new int[3][];
//İkinci boyutta bulunan her elemanda bir dizi olduğuna göre
//onlarıda new komutu ile ayarlamamız gerekir.
intCokBDizi[0] = new int[2];
```

```
intCokBDizi[1] = new int[4];
intCokBDizi[2] = new int[3];
//Dizi elemanlarına değer ataması
intCokBDizi[0][0] = 13;
intCokBDizi[0][1] = 15;
intCokBDizi[1][0] = 21;
intCokBDizi[1][1] = 29;
intCokBDizi[1][2] = 29;
intCokBDizi[1][3] = 23;
intCokBDizi[2][0] = 39;
intCokBDizi[2][1] = 39;
intCokBDizi[2][2] = 33;
```

Bir önceki anlattığımız düzensiz çok boyutlu dizide for döngüsü kullanarak elemanlara tek tek ulaşmıştık. Düzensiz dizilerde elemanlara tek tek ulaşmak istediğimizde for döngüsü yerine foreach döngüsü kullanıyoruz.

foreach Döngüsü:

```
foreach (int[] row in intCokBDizi)
{
foreach (int col in row)
{
Console.WriteLine(col);
}
Console.WriteLine("---------");
}
```

Ekran Çıktısı

```
13
15
---------
21
29
29
23
---------
39
39
33
---------
```

Yukarıda anlattıklarımızın ışığında 3 boyutlu düzenli ve düzensiz dizileri nasıl tanımlarız ve bunlara nasıl değer ataması yaparız bir örnekle anlatalım.

```
//3 boyutlu düzenli dizi tanımlama
int[,,] intUcBDizi = new int[2,3,2];
//3 boyutlu düzenli diziye değer atama
intUcBDizi[0, 0, 0] = 1;
intUcBDizi[0, 0, 1] = 34;
intUcBDizi[1, 1, 0] = 156;
//3 boyutlu düzensiz dizi tanımlama
int[][][] intUcBDuzensizDizi = new int[2][][];
//3 boyutlu düzensiz dizinin alt elemanlarını ayarlama
intUcBDuzensizDizi[0] = new int[2][];
intUcBDuzensizDizi[0][0] = new int[3];
intUcBDuzensizDizi[0][1] = new int[4];
intUcBDuzensizDizi[1] = new int[3][];
```

```
intUcBDuzensizDizi[1][0] = new int[2];
intUcBDuzensizDizi[1][1] = new int[5];
intUcBDuzensizDizi[1][2] = new int[3];
//3 boyutlu düzensiz diziye değer atama
intUcBDuzensizDizi[0][0][0] = 129;
intUcBDuzensizDizi[0][0][1] = 65;
intUcBDuzensizDizi[0][1][0] = 119;
intUcBDuzensizDizi[0][1][1] = 35;
intUcBDuzensizDizi[1][1][1] = 30;
intUcBDuzensizDizi[1][2][1] = 37;
```

Örneklerimizde hep integer tipinde veri tutan dizilerden bahsettik. Örneklerle string değer tutan dizilerin nasıl kullanıldığını basitçe anlatalım.

```
string[] isim = new string[4];
```

Dizi elemanlarının değerlerini oluşturma anında vermek

```
string[] isim = new string[4]{"Sabri","Seher","Ali","Metin"};
//veya
string[] isim2 = new string[] { "Sabri", "Seher", "Ali", "Metin" };
//veya
string[] isim3 = { "Sabri", "Seher", "Ali", "Metin" };
//veya tanımlama ve değer oluşturma işlemini ayrı satırlarda yapma
string[] isim4;
isim4 = new string[4] { "Sabri", "Seher", "Ali", "Metin" };
```

İlk oluşturma anında elemanlara değer verme işlemini 2 boyutlu dizilerde de aşağıdaki gibi yapabiliriz.

```
string[][] isimListesi = { new string[] { "Sabri", "Seher" }, new
 string[] { "Ahmet", "Mehmet", "Zuhal", } };
```