SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ

SAKARYA MESLEK YÜKSEKOKULU

BİLGİSAYAR PROGRAMCILIĞI PROGRAMI

Algoritma ve Programlamaya Giriş Ders Notları

Ders 13: Programlamada Dizi Kavramı

Diziler Neden Kullanılır?

Boyutlarına Göre Diziler

Tek Boyutlu Diziler

Çok Boyutlu Diziler

Hazırlayan

Öğr. Gör. Dr. İsmail ÖYLEK

Sakarya

2024

Diziler ve Dizi İşlemleri

Giriş

Programlama dilleri verileri geçici ve kalıcı olarak saklayabilmek için birtakım imkânlar sunmaktadırlar. Programın çalışması esnasında sık ihtiyaç duyulabilecek ve farklı değerlerden oluşması muhtemel veriler çoğunlukla değişkenler üzerinde saklanırken kalıcı olması istenen ve genellikle sistemli bir şekilde saklanması gereken veriler harici veritabanlarında depolanır. Verilerin hızlı yönetimi için kullanılan değişkenler bilindiği üzere tek bir veriye ev sahipliği yapabilmekte olup yeni bir değer atanması sonucu eski değer kaybedilmektedir. Aynı türden olan çok sayıda veriyi değişken sistemi ile geçici olarak depolamak için ise **dizi değişkenler (*array*)** kullanılmaktadır. İndis (index) denilen sıralı yapı sayesinde aynı verinin alternatiflerini saklayabilmesi, döngüler yardımıyla indislere ulaşıp içeriğindeki veriler üzerinde istenilen işlemlerin gerçekleştirebilmesi üstelik bu işlemlerin hızlı bir şekilde uygulayabilmesi gibi özelliklerinden dolayı diziler programcıların vazgeçilmez kurtarıcılarıdır.

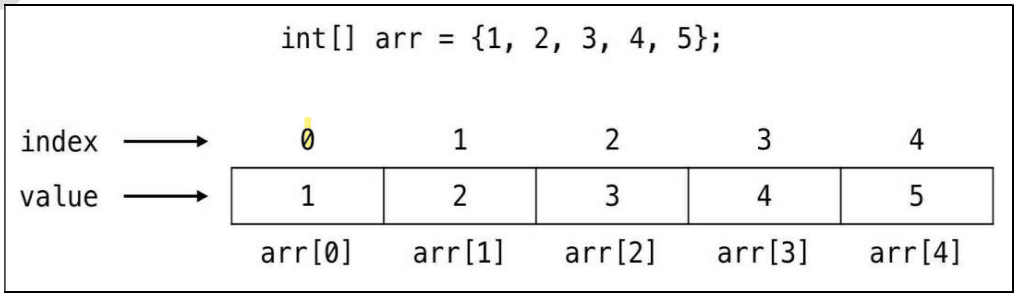
Programlama dillerinin çok eski versiyonlarından bu yana kullanılmakta olan diziler her ne kadar vazgeçilmez olsalar da bazı kısıtlarından dolayı benzer mantıkla çalışabilen muadilleri ortaya çıkmıştır. Visual Studio ailesinde kullanılan **Koleksiyonlar (*Collections*)** diziler gibi indis mantığıyla birçok veriyi depolayıp üzerinde işlem yapmaya imkân tanımaktadır. Hatta bazı nesnelerin özellik olarak barındırması gereken birtakım değerleri, dizi veya koleksiyon formatında saklanmaktadır. Bu ders dokümanında koleksiyonlar konusu işlenmeyecek olup diziler konusu ve diziler ile birlikte kullanılan bazı metot ve özellikler işlenecektir.

Diziler (Arrays)

Giriş bölümünde aynı türden çok fazla değeri tutabilen veri alanları olarak tanımlanan diziler, tek boyutlu ve çok boyutlu olarak tanımlanabilmektedir. Aşağıda boyutlarına göre diziler ve devamında dizilerde kullanılan bazı metotlara yer verilmiştir.

a) Boyutlarına Göre Diziler

1.1. Tek Boyutlu Diziler: Dizi yapısının en yalın ve temel hali olan tek boyutlu dizileri bir yük treninin vagonlarına benzetebiliriz. Genellikle benzer türden yükleri taşıyan yük vagonları, varsayılan olarak baştan sona doğru nizami olarak doldurulması gerekse de arada dolup boşalan yüklerden dolayı zaman zaman düzensiz dolulukta bulunabilirler. Tek boyutlu diziler de benzer mantıkla işlemektedir. Aşağıdaki görselde ismi `arr` olan `int[]` dizisi 5 vagondan oluşan bir katara benzemektedir. İndis değerleri 0-4 arasında değişen `arr` dizisinin sırasıyla 1-5 arasındaki tamsayıları sakladığı ve her bir sayıya ulaşmak için gerekli adresler alt kısımda görülmektedir.



Şekil 1. *Tek boyutlu dizi yapısı.*

Tek boyutlu bir dizinin oluşturulması, değer atanması ve değer okunması aşağıda gösterilmiştir.

```
char[] katar = new char[20]; //Dizi oluşturma. Yöntem 1
int[] tekboyutdizi = {1, 2, 3, 4, 5}; //Dizi oluşturma. Yöntem 2
katar[0] = 'a'; //Değer atama.
tekboyutdizi[0] = 10; //Değer atama.
Console.Write(tekboyutdizi[4]); //Değer kullanma (okuyup ekrana yazma)
```

1.2. Çok Boyutlu Diziler: 2 veya daha fazla boyuta sahip olarak oluşturulan diziler çok boyutlu dizi kategorisine girer. Normal bir tabloyu veya bir Excel çalışma sayfasındaki satır ve sütunlardan oluşan hücre bölmelerini 2 boyutlu dizilere örnek gösterebiliriz. Çok gözlü emanet dolaplarını da 2 boyutlu dizilere örnek verebiliriz ancak dolabın her bir gözü birkaç bölmeye ayrılmışsa 3 boyutlu dizi söz konusu olacaktır. Her bir boyut diğer boyutu/boyutları çoklayacağı için dizinin eleman kapasitesi boyutların eleman kapasitelerinin çarpımına eşit olur.

Çok boyutlu diziler çoğunlukla düzgün dikdörtgen (*matrix: matris*) benzeri yapıda olabileceği gibi düzensiz (*Jagged arrays*) yapıda da bulunabilirler. Aşağıda günlük hayatta kullandığımız ve 2 - 3 boyutlu dizilere benzeyen bazı yapılar görülmektedir.

Şekil 2. *Çok boyutlu diziler çok gözlü ve bölmeli çekmecelere sahip dolaplara benzetilebilir.*

Visual Studio'da çok boyutlu diziler tanımlanırken boyut sayısını belli edecek şekilde virgül kullanılır. Boyut sayısı arttıkça dizinin eleman saklama kapasitesi ile birlikte karmaşıklığı da artacaktır. 2 ve 3 boyutlu dizilerin oluşturulması, değer atanması ve değer okunması işlemleri aşağıda gösterilmiştir.

```
int[,] sayi1 = new int[2, 3]; //2D Dizi oluşturma.

int[,] sayi2 = new int[2, 3] { { 1, 2, 3 }, { 4, 5, 6 } }; //2D Dizi ve eleman atama.

Console.Write(sayi2[1, 2]);//sayi2 dizisinin son elemanını yazdırma.

int[,,] sayi3 = new int[4, 3, 2] { { { 11, 12 }, { 13, 14 }, { 15, 16 } }, { { 21, 22
}, { 23, 24 }, { 25, 26 } }, { { 31, 32 }, { 33, 34 }, { 35, 36 } }, { { 41, 42 }, {
43, 44 }, { 45, 46 } } }; //3D Dizi oluşturma ve eleman atama.
```

sayi3 isimli dizi 3 boyutlu olarak oluşturulmuş ve aynı satırda indislerine eleman ataması yapılmıştır. Bu dizinin 4 satırı, her satıda 3'er sütunu ve sütunların 2'şer bölmelik derinliği bulunmaktadır. Eleman kapasitesi ise 4x3x2=24'tür. Dizilerin satır ve sütunlarında yer alan sayılar aşağıdaki tabloda görülmektedir. 2 eleman kapasitesine sahip 3. boyuttan (derinlik) dolayı tablonun hücrelerinde ikişer sayı bulunmaktadır.

sayi3 dizisi	1. Sütun	2. Sütun	3. Sütun
1. Satır	11,12	13,14	15,16
2. Satır	21,22	23,24	25,26
3. Satır	31,32	33,34	35,36
4. Satır	41,42	43,44	45,46

Bu verilere aşağıdaki gibi kodlamayla erişilebilir.

```
Console.Write(sayi3[0, 0, 0]); //İlk satırın ilk sütunundaki ilk veri
Console.Write(sayi3[0, 2, 0]); //İlk satırın son sütunundaki ilk veri
Console.Write(sayi3[1, 2, 0]); //2. satırın son sütunundaki ilk veri
Console.Write(sayi3[2, 1, 1]); //3. satırın orta sütunundaki son veri
Console.Write(sayi3[3, 2, 1]); //Son satırın son sütunundaki son veri
```

b) Dizilerle Birlikte Kullanılan Bazı Metotlar

- **Array.Resize( ):** Diziler oluşturulurken veya kullanım öncesinde boyutları belirlenir. Fakat bu boyut belirleme işlemi sonradan değiştirilemeyecek kalıcı bir atama değildir. Örneğin aşağıdaki gibi bir işlemle 5 eleman kapasitesi bulunan **tekboyut** dizisinin boyutu sonradan 10 olarak değiştirilebilir.

```
int[] tekboyut = { 1, 2, 3, 4, 5 };
tekboyut = new int[10];
Console.Write(tekboyut[0]);
```

Yukarıdaki kodlarda, 2. satırda yapılan değişiklik dizinin boyutunu 10 olarak güncellemekle beraber dizi yeniden oluşturulduğundan içeriğindeki verileri de boşaltacaktır. Böylece son satırda görülen dizinin ilk elemanını yazdırma işlemi başarısız olacaktır. Eğer dizide saklanan değerlerin kaybedilmemesi gerekiyorsa o zaman new ifadesi yerine Array.Resize( ) metodu uygulanmalıdır. Resize() metodu başına Array ifadesi alarak çağrılır. Hangi dizi üzerinde boyut değişikliği yapılacaksa ilk parametre olarak önüne yazılan ref ifadesi ile dizi ismi belirtilir, ikinci parametre olarak da istenilen güncel dizi boyutu yazılarak komut tamamlanır. Aşağıdaki kodlarda eleman atamasıyla 5 elemanlı olarak oluşturulan dizi ikinci satırda Resize() metodu ile 3 eleman kapasitesine düşürülmüştür. Eleman sayısı kırpıldığı için son iki indisindeki veriler doğal olarak kaybedilir fakat ilk 3 indiste herhangi bir veri kaybı oluşmaz.

```
int[] tekboyut = { 1, 2, 3, 4, 5 };
Array.Resize(ref tekboyut, 3);
Console.Write(tekboyut[1]);
```

- **SetValue( ):** Dizi indislerini set etmek yani eleman ataması yapmak için kullanılan metottur. Dizi isminden sonra çağrılan SetValue() metodunun ilk parametresinde atanacak değer, ikinci parametresinde ise hangi indise atama yapılacağı belirtilir. Çok boyutlu dizi ile çalışılıyorsa virgüllerle ayrılmış indis numaraları yazılır.

- **GetValue( ):** Dizinin indislerinde yer alan elemanlara erişmek için kullanılır. Dizi isminden sonra çağrılan GetValue() metodunun indis numarası isteyen tek parametresi vardır. Çok boyutlu dizi ile çalışılıyorsa virgüllerle ayrılmış indis numaraları yazılır.

```
string[] adsoy = new string[5]; //adsoy isimli, 5 elemanlı bir string dizisi
adsoy.SetValue("Mehmet Ali Yılmaz", 0); //ilk indisteki veri: Mehmet Ali Yılmaz
adsoy[1] = "Elif Ayyıldız"; //ikinci indise atanan veri: Elif Ayyıldız
Console.Write(adsoy.GetValue(0)); //ilk indisteki veriyi al
Console.Write(adsoy[1]); //ikinci indisteki veriyi al
```

Yukarıda ***SetValue()*** ve ***GetValue()*** metotları tek örnekte gösterilmiştir. Aynı işlemler çoğunlukla tercih edilen köşeli parantez **[ ]** operatörü ile de gerçekleştirilmiştir.

- **Join( ):** Parametre olarak verilen dizinin içerisinde bulunan tüm elemanları, aralarına ayırıcı bir karakter ekleyerek birleştirir. Dizi elemanları, belirtilen ayırıcı karakter eklenerek geriye `string` türünde döner. Ayırıcı (*seperator*) parametre olarak herhangi bir değer verilmezse dizi elemanları yan yana birleşik yazılarak geriye `string` türünde döner. `String` sınıfından bir metot olduğu için `String.Join()` yazılarak çağrılır.

- **Copy( ):** Dizi içeriğini başka bir dizi indislerine kopyalamak için kullanılır. Array sınıfı altında çağırılabilir. Sırasıyla ***Kaynak Dizi Adı, Hedef Dizi Adı, Kopyalanacak Eleman Adedi*** olmak üzere 3 parametre alır.

- **Contains( ):** Kendisini çağıran dizi içeriğinde aranan bir değer var mı, yok mu arama yapar. Aranan eleman bulunursa `true`, bulunamazsa `false` değer üretir. Aranan değeri isteyen tek parametresi vardır.

- **IndexOf( ):** Dizi içinde baştan arama yaparak geriye indis numarası döndüren metottur. Zorunlu iki parametreye ek olarak opsiyonel 1 parametresi daha vardır. Bunlar sırasıyla ***Arama Yapılacak Dizi Adı, Aranan Değer*** ve opsiyonel parametre olan ***Aramanın Başlayacağı İndis Değeri***'dir. Opsiyonel parametre girilmezse arama 0. indisten başlar. Aranan değer birden fazla bulunsa da ilk bulunanın indis numarası döndürülür. Array sınıfı altında çağırılabilir.

- **LastIndexOf( ):** Dizi içinde sondan arama yaparak geriye indis numarası döndüren metottur. Zorunlu iki parametreye ek olarak opsiyonel 1 parametresi daha vardır. Bunlar sırasıyla ***Arama Yapılacak Dizi Adı, Aranan Değer*** ve opsiyonel parametre olan ***Aramanın Başlayacağı İndis Değeri***'dir. Opsiyonel parametre girilmezse arama en son indisten başlar. Aranan değer birden fazla bulunsa da en sonda bulunanın indis numarası döndürülür. Array sınıfı altında çağırılabilir.

```
//Join, Copy, Contains, IndexOf, LastIndexOf örnekleri
string[] adsoy = new string[5] { "Ali", "Veli", "Ayşe", "Hikmet", "Elif" };
//adsoy içeriğini virgül ile bağlayıp isimler değişkenine ata
string isimler = String.Join(", ", adsoy);
string[] names = new string[3];
//adsoy dizisinin ilk 3 elemanını names dizisine kopyala
Array.Copy(adsoy, names, 3);
//names dizisinde Hikmet verisini ara. sonuc: false
bool sonuc = names.Contains("Hikmet");
int ara1, ara2;
//adsoy dizisinde 2. indisten itibaren Ali verisini ara. ara1: -1
ara1 = Array.IndexOf(adsoy, "Ali", 2);
//adsoy dizisinde sondan itibaren Ali verisini ara. ara1: 0
ara2 = Array.LastIndexOf(adsoy, "Ali");
```

Kaynaklar

1. Volkan AKTAŞ, Her Yönüyle C# 5.0, Kodlab Yayıncılık, 2013, İstanbul.

2. https://tr.wikibooks.org/wiki/C_Sharp_Programlama_Dili/Diziler, Erişim Tarihi: 16.07.2020.