

Bu dokümanda düzenli değişkenler yani diziler konu edilmiştir. Genel yapı çizilmiş olup ayrıntılar hakkında az örnek verilmiştir. Anlamadığınız kısımda lütfen yapay zekadan ve internetten yardım alın ve araştırın.

## Diziler

Sıralı değişkenler, düzensiz diziler, matrisler ve metin değişkeni

Oğuzhan Karagüzel

01.12.2024

## İçindekiler

ŞEKİLLER DİZİNİ.....	3
C# PROGRAMLAMA DİLİNE GİRİŞ: DİZİLER (ARRAYS) VE KULLANIMI .....	4
Diziler Nedir? .....	4
Dizi Tanımlama ve Kullanımı .....	4
Dizi Tanımlama Örneği.....	4
Dizi Elemanlarına Erişim .....	5
`[]` Operatörü Nedir?.....	6
Okuma .....	6
Yazma.....	6
Dizi Tanımlarken Nelere Dikkat Edilmeli? .....	8
Değişkeni Çağırırken Nelere Dikkat Edilmeli? .....	8
NEW ANAHTAR KELİMESİ .....	9
İlk Değer Ataması.....	10
1. Diziye Tanımlarken Değer Atama:.....	11
2. `new` Anahtar Kelimesi ile Değer Atama:.....	11
3. `new` Anahtar Kelimesi ile Boyut ve Değer Atama:.....	11
4. Varsayılan Değerlerle Başlatma: .....	12
STRING VE CHAR ARRAY ÜZERİNE DETAYLI İNCELEME .....	12
`char[]` ve `string` ilişkisi.....	13
String İçerisindeki Her Bir Karaktere Erişim.....	14
`string[]` Dizisi.....	14
Jagged Array ve String Karakterlerine Erişim.....	15
MATRİSLER VE KULLANIMI.....	18
Matris Nedir?.....	19
C# Dilinde Matris Tanımlama.....	19
Matris Elemanlarına Erişim.....	20
Matris Tanımlama ve İlk Değer Ataması .....	20
Matris ve Jagged Array Farkları .....	21
Jagged Array ve Matris Elemanlarına Erişim .....	21
Matris ve Jagged Array Tanımlarken Dikkat Edilmesi Gerekenler.....	21
Matris ve Jagged Array Örnekleri .....	22
Matris Örneği.....	22
Jagged Array Örneği.....	23
DİZİ ELEMANLARINA ADRESLERİ İLE DOĞRUDAN ULAŞMA ÖRNEĞİ.....	24

ÖZET VE SONUÇ.....	25
ÖDEV.....	26
TEST .....	26
CEVAP ANAHTARI.....	29

Öğuzhan KARAGÜZEL

## ŞEKİLLER DİZİNİ

Şekil 1 Dizi tanımlama.....	4
Şekil 2 Dizi Elemanlarına Erişim .....	6
Şekil 3 New Anahtar Kelimesi İle Değişken Tanımlama .....	10
Şekil 4 Diziyi Tanımlarken Değer Atama .....	11
Şekil 5 New Anahtar Kelimesi ile Değer Atama .....	11
Şekil 6 New Anahtar Kelimesi ile Boyut ve Değer Atama .....	11
Şekil 7 Varsayılan Değerlerle Başlatma .....	12
Şekil 8 C#'ta char[] tanımlama .....	13
Şekil 9 String Değişken Tanımlama .....	13
Şekil 10 String Değişkenin Karakterlerine Ulaşma .....	14
Şekil 11 String Dizisi .....	14
Şekil 12 String Dizisi Eleman Erişimi .....	15
Şekil 13 jagged Array Tanımlama .....	15
Şekil 14 jaggggeeeeddddDizi .....	16
Şekil 15 Jagged Dizi Erişimi .....	16
Şekil 16 String Dizisi Tek Karakter Ulaşım .....	16
Şekil 17 String Dizisi Tek Karakter Ulaşım Programı Çıktısı .....	17
Şekil 18 String Değişkenine Char[] olarak Yaklaşım .....	17
Şekil 19 String Değişkenine Char[] olarak Yaklaşım Programı Çıktısı .....	18
Şekil 20 Matris Tanımlama.....	19
Şekil 21 Matris Elemanlarına Değer Atama .....	19
Şekil 22 Matris Elemanlarına Erişim .....	20
Şekil 23 Matris Tanımlama ve İlk Değer Ataması .....	20
Şekil 24 5 Boyutlu Matris.....	24
Şekil 25 Byte Array Adres İle Elemanlara Teker Teker Ulaşma .....	25
Şekil 26 Byte Array Adres İle Elemanlara Teker Teker Ulaşma Program Çıktısı.....	25

# C# PROGRAMLAMA DİLİNE GİRİŞ: DİZİLER (ARRAYS) VE KULLANIMI

## Diziler Nedir?

C# programlama dilinde, birden fazla değeri tutmak ve bu değerlere kolayca erişebilmek için kullanılan veri yapılarından biri dizilerdir. Dizi (Array), aynı türdeki birden fazla veriyi bir arada tutan bir koleksiyondur. Diziler, verileri ardışık bir şekilde bellekte depolar ve her bir elemana indeks numaralarıyla erişim sağlar. Bu özellikleri, büyük veri kümeleriyle çalışırken düzenli ve etkili bir yöntem sunar.

Diziler, programlama sürecinde birçok farklı durumda kullanılır. Örneğin, bir sınıftaki öğrencilerin notlarını tutmak veya bir alışveriş sepetindeki ürünlerin fiyatlarını saklamak için diziler kullanılabilir. Diziler, bir veri kümesi üzerinde işlem yapmayı daha kolay hale getirir ve veriler üzerinde döngülerle işlem yaparken de oldukça faydalıdır.

## Dizi Tanımlama ve Kullanımı

Bir dizi oluştururken, dizi elemanlarının türü ve boyutu belirlenmelidir. Dizin boyutu, dizi oluşturulduktan sonra değiştirilemez. Bu nedenle, bir dizi kullanmadan önce ihtiyacınıza uygun boyutu belirlemek önemlidir. Bazı yöntemler kullanılarak aslında kolaylıkla dizinin boyutunu manipüle etmekteyiz. Ancak bu fazladan işlem yükü demektir. Daha açık konuşmak gerekirse; 5 elemanlı bir dizi oluşturduktan sonra eğer altıncı elemana ihtiyaç duyarsanız, 6 elemanlı bir dizi oluşturup diğer dizinin 5 elemanını ve sizin 6. elemanınızı bu dizinin içine atarak bunu yapabilirsiniz. C# programlama dilinde bu gibi kodlama süreçlerini kısaltan metotlar bulunmaktadır. Birçok modern programlama dilinde bulunmaktadır. Ancak her zaman en iyi yol tanımladığınız bir değişkeni en etkin biçimde kullanmaya çalışmaktır.

C# dilinde bir dizi tanımlamak için `[]` operatörü kullanılır.

## Dizi Tanımlama Örneği

Bir tam sayı dizisi tanımlamak ve bu diziyi beş eleman atamak için şu şekilde bir kod kullanabiliriz:

```
static void Main(string[] args)
{
    int[] sayilar = new int[5];
}
```

Şekil 1 Dizi tanımlama

Yukarıdaki kodda, `int` türünde beş eleman içerebilecek bir dizi tanımlıyoruz. `sayilar` adını verdiğimiz bu dizi, bellekte beş adet `int` türünde veri saklayabilecek bir alan oluşturur. Bu dizinin her bir elemanına indeks numarası kullanarak erişim sağlanabilir.

Bir önceki dersten hatırlıyor olmalısınız. Eğer bir önceki dersi incelemediyseniz ilk olarak o dersi incelemeniz gerekmektedir. Burada anlamamız gereken asıl durum şudur. "int" veri tipinin 4 byte olduğunu biliyorsunuz.

Eğer burada değişkenimizi "int sayi = 10" şeklinde tanımlasaydık, 10 sayısı ikilik tabana çevrilip bu 4 byte'a sayının değeri yazılacaktır. En anlamsız byte'ın adresi de "sayi" değişken adı ile temsil edilecektir. Bizler "sayi" adlı değişkeni çağırdığımızda en anlamsız byte'ın adresini çağırmış olurduk. .Net ve işletim sistemi ortak bir biçimde çalışarak bu değişkenin "int" türünde olduğunu anlar ve bu adres değerine eklemeler yaparak sayının tamamını elde eder. "sayi" diye temsil etmemizin sebebinin ise bu adresin işletim sistemi tarafından otomatik atanması olduğunu biliyorsunuz. .Net ve işletim sistemi şu şekilde çalışarak tüm sayıyı elde eder:

Sayı "int" türünde herhangi bir adres olmak koşulu ile

- "sayi" ilk 8 bit veya 1. byte,
- "sayi + 1" ikinci 8 bit veya 2. byte,
- "sayi + 2" üçüncü 8 bit veya 3. byte,
- "sayi + 3" dördüncü 8 bit veya 4. byte,

Toplamda 32 bit elde edildikten sonra sayı geriye döndürülür ve çağırıldığı yere yazılır.

Biz "[]" "dizi" operatörü ile değişkenimizden birden fazla oluşturacağımızı söylüyoruz. .Net ve işletim sistemi bu değişkenin adreslerini birbirinin ardı ardına olacak şekilde RAM'de alan açarak değişkeni oluşturur.

Ancak kaç tane açacağını programımızda belirtmemiz gerekmektedir.

Örnek olarak "int[] sayılar = new int[5];" 5 sayısı, 5 tane "int" türündeki değişkeni ardı ardına oluşturacağımızı söylemiş oluruz. "int" değişkeni 4 byte alan kapladığı için "int[5]" ile tanımlanan değişken 4\*5'ten 20 byte alan kaplamaktadır ve her bir byte birbirinin ardına dizilecek şekilde konumlandırılır.

Bir önceki derste "int sayi = 10;" değişkeninde "sayi" değişken adının temsil ettiği adresin, 4 byte'ta en anlamsız byte'ın adresinin olduğunu söylemiştik.

Burada da durum değişmemektedir. "sayılar" değişkeni "int[5]" türünde olduğu için ilk sayının en anlamsız byte'ının adresini tutmaktadır. Bu sayılara erişirken bu durumu iyi kavramalıyız.

## Dizi Elemanlarına Erişim

Bir dizideki elemanlara erişmek için indeks numaraları kullanılır. C# dilinde dizilerin indeks numaraları `0`'dan başlar. Bu, bir dizinin ilk elemanının indeks numarasının `0`, ikinci elemanının indeks numarasının `1` olduğu anlamına gelir.

Örneğin, yukarıda tanımladığımız `sayılar` dizisine değer atamak için şu şekilde bir kod kullanabiliriz:

```
static void Main(string[] args)
{
    int[] sayilar = new int[5];
    sayilar[0] = 10;
    sayilar[1] = 20;
    sayilar[2] = 30;
    sayilar[3] = 40;
    sayilar[4] = 50;
}
```

Şekil 2 Dizi Elemanlarına Erişim

Yukarıdaki kodda, `sayilar` dizisinin her bir elemanına belirli değerler atıyoruz. `sayilar[0]` ifadesi, dizinin ilk elemanını temsil eder ve bu elemanın değeri `10` olarak atanmıştır. Benzer şekilde, dizinin diğer elemanlarına da değerler atanmaktadır. İlk elemanın değerinin neden 0 olduğuna ileride değinilecektir.

## `[]` Operatörü Nedir?

`[]` operatörü, dizilerle çalışırken kullanılan ve diziye erişimi sağlayan bir operatördür. Bu operatör, dizinin belirli bir indeksindeki elemanı okumak veya o elemana değer atamak için kullanılır. Örneğin, `sayilar[2]` ifadesi, `sayilar` dizisinin üçüncü elemanını temsil eder.

`[]` operatörünü kullanarak dizinin herhangi bir elemanına hem okuma hem de yazma işlemleri yapılabilir:

### Okuma

```
int deger = sayilar[2];
```

Yukarıdaki kodda, `sayilar` dizisinin üçüncü elemanının değeri `deger` değişkenine atanır.

### Yazma

```
sayilar[2] = 35;
```

Bu kod, `sayilar` dizisinin üçüncü elemanının değerini `35` olarak günceller.

Burada kafanızın karışması normal bir durum. Neden ilk elemanın indeksi `0`? 3. Elemanın indeksi neden 2. Değişkene erişmek için neden `[]` operatörünü kullanıyoruz? Hadi detaylandıralım.

Tekrar ve Tekrar; "int sayi = 10;" sayi = en anlamsız byte'ın adresi.

Programın herhangi bir yerinde "sayi" diye çağırıldığında bu adrese git ve oku denilir. Artık nasıl okunduğunu öğrendiğinizi var sayıyorum.

"int[] sayilar = new int[5];" ise 20 byte'lık bir veri tipidir ve ilk byte'ın yani en anlamsız byte'ın adresi sayilar ile temsil edilir.

Bu dizinin elemanlarına ulaşmak için formülümüz şudur:

**(adres + (değişken tipinin boyutu \* indeks) ) + değişken tipinin ulaşılması istenilen byte'ının sayısı.**

Bu durum bize ilk elemanın indeks değerinin neden `0` olduğunu da açıklar.

Hadi "sayilar" değişkenini inceleyelim. `sayilar` 5 elemanlı `int` türünde bir dizi değişkeni. Temsil ettiği adres ilk elemanın en anlamsız byte'ının adresi.

Bu durumda bu formül ile ilk elemana ulaşmaya çalışırsak;

- `sayilar` herhangi bir adres. `int` 4 byte ve indeks `0` ise; ilk elemana ulaşmak için 4 byte'a ulaşmak lazım. Bu durumda:

- `(sayilar + (4\*0)) + 0 = sayilar + 0` = ilk 8 bit'i veya 1. byte'ı;
- `(sayilar + (4\*0)) + 1 = sayilar + 1` = ikinci 8 bit'i veya 2. byte'ı;
- `(sayilar + (4\*0)) + 2 = sayilar + 2` = üçüncü 8 bit'i veya 3. byte'ı;
- `(sayilar + (4\*0)) + 3 = sayilar + 3` = dördüncü 8 bit'i veya 4. byte'ı;

Üçüncü elemana ulaşmaya çalışalım. Bu durumda indeks iki olacaktır. Hadi bakalım formül doğru mu işleyecek;

- `(sayilar + (4\*2)) + 0 = sayilar + 8` = üçüncü elemanın ilk 8 bit'i veya 1. byte'ı
- `(sayilar + (4\*2)) + 1 = sayilar + 9` = üçüncü elemanın ikinci 8 bit'i veya 2. byte'ı
- `(sayilar + (4\*2)) + 2 = sayilar + 10` = üçüncü elemanın üçüncü 8 bit'i veya 3. byte'ı
- `(sayilar + (4\*2)) + 3 = sayilar + 11` = üçüncü elemanın dördüncü 8 bit'i veya 4. byte'ı

"sayilar" 1. elemanın 1. byte'ı

"sayilar + 1" 1. elemanın 2. byte'ı

"sayilar + 2" 1. elemanın 3. byte'ı

"sayilar + 3" 1. elemanın 4. byte'ı

"sayilar + 4" 2. elemanın 1. byte'ı

"sayilar + 5" 2. elemanın 2. byte'ı

"sayilar + 6" 2. elemanın 3. byte'ı



"sayılar + 7" 2. elemanın 4. byte'ı

"sayılar + 8" 3. elemanın 1. byte'ı

Teker teker kontrol ettiğimizde formülümüz doğru işlemektedir.

Artık dizi tanımlama ve dizinin elemanlarına ulaşmanın nasıl olduğunu öğrendiğinizi varsayıyorum. Kısaca:

- "değişkenin tipi [dizi operatörü] değişken adı = new değişkenin tipi [dizinin uzunluğu]" tanımlama
- "değişken adı [indeks numarası]" ulaşma.

Aklınızda kalan son sorunun `new` anahtar kelimesi olduğunu biliyorum. Dokümanın sonlarına doğru bir başlıkta bu anahtar kelimeye bir bakış atacağız.

## Dizi Tanımlarken Nelere Dikkat Edilmeli?

Dizi tanımlarken dikkat edilmesi gereken bazı önemli noktalar vardır:

- 1. Veri Türü:** Dizinin hangi türde verileri saklayacağını belirtmelisiniz. Bu, dizinin her bir elemanının aynı türde olmasını sağlar. Örneğin, `int[]`, `string[]`, `double[]` gibi farklı türlerde diziler tanımlanabilir.
- 2. Boyut Belirleme:** Dizinin boyutunu tanımlarken dikkatli olmalısınız. Dizinin boyutu, dizi tanımlandıktan sonra değiştirilemez, bu nedenle dizinin kaç eleman tutacağını önceden belirlemek önemlidir.
- 3. Bellek Kullanımı:** Dizi tanımlarken, bellekte ayrılacak olan alanın büyüklüğüne dikkat etmek gerekir. Büyük boyutlu diziler tanımlarken, programın bellek tüketimini ve performansını göz önünde bulundurmalısınız.

## Değişkeni Çağırırken Nelere Dikkat Edilmeli?

Bir dizinin elemanına erişmek veya onu kullanmak istediğinizde bazı noktalara dikkat etmeniz gerekir:

- 1. Geçerli İndeks Kullanımı:** Dizilerde geçerli bir indeks kullanmak önemlidir. Dizinin boyutunun dışında bir indeks kullanmak, `IndexOutOfRangeException` hatasına yol açar. Örneğin, `sayılar` dizisinin `5`. elemanı yoktur, bu nedenle `sayılar[5]` ifadesi bir hata oluşturacaktır. Dikkat! Dizi 5 elemanlıdır. Son elemanın indeks değeri 4'tür.

- 2. Dizinin Başlangıç Değerleri:** C# dilinde, bir dizi oluşturulduğunda elemanlarına varsayılan değerler atanır (`new` anahtar kelimesinde buna daha detaylı değineceğiz). Örneğin, `int` türünde bir dizi

oluşturulduğunda, dizinin tüm elemanlarının başlangıç değeri `0` olur. Bu durum, diziye kullanmadan önce tüm elemanların başlangıç değerlerini kontrol etmeniz gerektiği anlamına gelir.

## NEW ANAHTAR KELİMESİ

C# programlama dili "object oriented programming" (OOP) ya da nesne yönelimli programlama dilidir. Nesne yönelimli programlamayı belki duymuş olabilirsiniz, ancak o konuya gelmemize daha çok var. C# programlama dili, C programlama dili tabanlıdır. Aslında bildiğiniz birçok programlama dili C tabanlıdır, yani bu programlama dilleri C programlama dili ile geliştirilmiştir. `new` anahtar kelimesi ise nesne tabanlı programlama ile ortaya çıkan bir ihtiyaçtan doğmuştur.

Şimdi size bir örnek vermek istiyorum. Bu örneği doğrudan vermek yerine "copilot" adlı yapay zekaya sordum ve oldukça güzel bir açıklama yaptı. Hadi inceleyelim.

**Soru;** C programlama dilinde `int c;` değişkeni oluştursam. Sonra `printf` ile `c`'yi hemen bir alt satırda ekrana yazdırsam. Çıktı ne olur. Bir hata alır mıyım?

**Cevap;** Eğer C programlama dilinde `int c;` değişkenini oluşturur ve ardından `printf` ile `c`'yi hemen bir alt satırda ekrana yazdırırsan, şöyle bir kod elde edersin:

```
#include <stdio.h>

int main() {
    int c;

    printf("%d\n", c);

    return 0;
}
```

Bu durumda, `c` değişkeni için bir değer atanmamış olur. Bu da, `c` değişkeninin içinde rastgele bir değer bulunabileceği anlamına gelir. Yani, `printf("%d\n", c);` satırı çalıştırıldığında ekranda herhangi bir belirli değeri görebilirsin, bu değer büyük ihtimalle anlamlı bir sayı olmaz.

Çoğu derleyici bu durumu tanımlanmamış davranış olarak değerlendirir ve bu nedenle hata mesajı almazsın. Ancak, programın çıktısı beklenmedik sonuçlar verebilir ve bu, programın kararsız çalışmasına neden olabilir.

Değişkenleri kullanmadan önce her zaman bir başlangıç değeri atamak iyi bir uygulamadır. Mesela:

```
#include <stdio.h>

int main() {
    int c = 0; // Başlangıç değeri atama

    printf("%d\n", c);

    return 0;
}
```

Bu şekilde, c değişkeni önceden belirlenmiş bir değerle başlatılmış olur ve programın çıktısı öngörülebilir hale gelir.

---

Bu konuya "Değişkenler ve Operatörler" dersinde kısaca değinmiştim. Bir değişken tanımladığınızda RAM'de bir alan açılır ve ilk byte'ın adresi size verilir, ancak bu alanda bir değer olup olmadığı kontrol edilmez. Eğer C programlama dilinde bir değişken oluşturur ve ilk değeri atamazsanız rastgele bir sayı ile karşılaşabilirsiniz. Bunu önlemek için ilk değerini atamalısınız.

`new` anahtar kelimesinin işlevi budur: ilk değerin atanmasıdır. Oluşturduğunuz değişkenin veya dizinin elemanlarının ilk değerinin atanmasıdır. Bu çoğu değişken tipi için `0`'dır. `int[] sayilar = new int[5]` ifadesinde her elemana (`5` eleman) `0` değeri atanır. Bu işlem `new` anahtar kelimesi ile yapılmaktadır.

Şimdi aklınızı karıştıran sorunun farkındayım.

"int sayi;" diye değişken tanımlarken ya da "int sayi = 10;" diye değişken tanımlarken "new" anahtar kelimesini kullanmadık. ancak dizileri tanımlarken "int[] sayilar = new int[5];" şeklinde değişken tanımlarken new anahtar kelimesini kullandık.

İlk olarak, `int sayi;` değişkenini tanımlarken `new` anahtar kelimesini kullanmasak bile arka planda bu süreç işlemeye devam eder, yani ilk değeri atanır. `int sayi` adlı değişkeni tanımladıktan sonra çağırmaya çalıştığınızda programınız hata verecek ve ilk değeri atamadığınızı belirtecektir.

İlk değeri doğrudan kendiniz atayabilirsiniz. Ya da şu şekilde de tanımlama yapabilirsiniz:

```
static void Main(string[] args)
{
    int sayi = new int();
}
```

Şekil 3 New Anahtar Kelimesi İle Değişken Tanımlama

`int sayi = new int();` Bu şekilde tanımlama yaptığınızda ilk değer `0` olarak atanacaktır ve artık programınız hata vermeyecektir.

**Aslında new anahtar kelimesinin tek görevi ilk değerleri atama değildir. Başka görevleride bulunmaktadır. Ancak buna nesne tabanlı programlamada değinilecektir.**

Bu konuyu anladığınızı düşünüyorum. `new` anahtar kelimesi, beklenmedik durumları önlemek için ilk değer atamasını yapmaktadır.

Şimdi ise aklınıza takılan sorunun `int()` olduğunu farkındayım. Bu konuya nesne tabanlı programlamada gireceğiz. Kısaca bahsetmek gerekirse, bu yapıcı (constructor) metod olarak adlandırılır ve nesne tabanlı programlama ile gelen bir durumdur. Bu konuya ileride daha ayrıntılı değineceğiz.

Peki, `new` anahtar kelimesi kullanılmadan ilk değer ataması yapılabilir mi? Hadi bir bakış atalım.

## İlk Değer Ataması

Dizilere ilk değer ataması yapmak, diziyi tanımlarken doğrudan elemanlara başlangıç değerleri vermek anlamına gelir. Bu, diziyi başlatırken her eleman için varsayılan bir değeri belirlemek istediğimizde kullanılır. İlk değer ataması, diziyi hızlıca başlatmak ve kullanıma hazır hale getirmek için oldukça kullanışlıdır. Aşağıdaki örneklerle dikkat edin bazı durumlarda new anahtar kelimesini kullanmamıza gerek yoktur. Bu durum arka planda işlenir. 1. örnekte atanan değerlerin 4 tane olduğu belirlenir ve dizinin boyutu 4 olarak atanır.

Dizilere ilk değer ataması yapmanın birkaç yolu vardır:

#### 1. Diziyi Tanımlarken Değer Atama:

```
static void Main(string[] args)
{
    int[] sayilar = { 5, 4, 9, 12 };
}
```

Şekil 4 Diziyi Tanımlarken Değer Atama

Bu örnekte `sayilar` adında bir dizi tanımlıyoruz ve dört elemanına doğrudan değer atıyoruz. Bu şekilde bir tanımlama, dizinin boyutunu otomatik olarak belirler ve `sayilar` dizisi `4` elemanlı bir dizi olarak oluşturulur.

#### 2. `new` Anahtar Kelimesi ile Değer Atama:

```
static void Main(string[] args)
{
    int[] sayilar = new int[] { 5, 4, 9, 12 };
}
```

Şekil 5 New Anahtar Kelimesi ile Değer Atama

Burada `new int[]` ifadesi kullanılarak dizi oluşturuluyor ve elemanlarına başlangıç değerleri atanıyor. Bu yöntem, dizi tanımlarken elemanların başlangıç değerlerini belirtmek için kullanılır.

Bu durumda “new” anahtar kelimesinin kullanımına gerek yoktur. Ancak “new” anahtar kelimesini kullanıyorsanız “int[]” ifadesini de yazmak zorundasınız. Yoksa programınız hata verecektir. Ancak bu şekilde ilk değer ataması yaptığınızda dizinin boyutunu belirtmenize de gerek yoktur. Otomatik olarak atanır.

#### 3. `new` Anahtar Kelimesi ile Boyut ve Değer Atama:

```
static void Main(string[] args)
{
    int[] sayilar = new int[4] {5, 4, 9, 12};
}
```

Şekil 6 New Anahtar Kelimesi ile Boyut ve Değer Atama

Bu örnekte `new int[4]` ifadesi ile dizinin boyutunu açıkça belirtiyoruz ve ardından elemanlara başlangıç değerlerini atıyoruz. Dizinin boyutu, başlangıçta belirttiğimiz değerlerle uyumlu olmalıdır. Bu örnekte `4` elemanlı bir dizi oluşturulmuştur ve bu elemanlara başlangıçta `5, 4, 9, 12` değerleri atanmıştır. Eğer 5 elemanlı deyip 4 eleman atarsanız hata alırsınız.

#### 4. Varsayılan Değerlerle Başlatma:

```
static void Main(string[] args)
{
    int[] sayilar = new int[4];
}
```

Şekil 7 Varsayılan Değerlerle Başlatma

Bu şekilde bir dizi tanımlandığında, elemanlarına otomatik olarak varsayılan değerler atanır. `int` türünde bir dizi olduğu için, tüm elemanların başlangıç değeri `0` olacaktır. Bu yöntem, belirli bir boyutta dizi oluşturmak, ancak elemanları daha sonra atamak istediğiniz durumlarda kullanılır.

Dizilere ilk değer ataması yaparken dikkat edilmesi gereken bazı noktalar vardır. Örneğin, dizi elemanlarının sayısı ile atanan değerlerin sayısı uyumlu olmalıdır. Aksi takdirde derleyici hatası alabilirsiniz. Ayrıca, `new` anahtar kelimesi kullanmadan doğrudan değer ataması yapmak, kodun okunabilirliğini artırabilir ve daha sade bir yazım sağlar.

Bu örnekler, dizilerle çalışırken ilk değer ataması yapmanın farklı yollarını göstermektedir. Dizilere başlangıçta değer atamak, kodun daha anlaşılır ve kullanışlı olmasına yardımcı olur. Bu yöntemleri kullanarak dizilerinizi ihtiyaçlarınıza göre şekillendirebilir ve hızlıca kullanıma hazır hale getirebilirsiniz.

Bu konuyu anladığınızı düşünüyorum. Şimdi yazılım için önemli bir konuya geliyoruz. `char[]` ve `string` değişken türüne bir bakış atalım.

Bu bölümde dizilere ilk değer ataması yapmanın yollarını ve bunun nasıl yapılabileceğini öğrendik. Dizilerin nasıl tanımlandığı, `new` anahtar kelimesinin nasıl kullanıldığı ve elemanlara nasıl başlangıç değeri atandığı konularına dair daha fazla sorunuz varsa bunu araştırın. Yapay zekâ ile sohbet edin. İlk derste önerdiğim kitaplarda bakın ya da kendi bulduğunuz kitapları inceleyin. Ya da en iyisi başka bir yazılımcı adayı bulun ve üzerine beyin fırtınası yapmaya başlayın. Merak etmeyin tam olarak doğruyu bulamasanız bile kesinlikle doğruya yaklaşacaksınız. Kesinlikle aklınızda çok daha iyi yer edecek. Hatalarınızı buldukça daha da iyi bir biçimde öğreneceksiniz.

## STRING VE CHAR ARRAY ÜZERİNE DETAYLI İNCELEME

C# dilinde, `string` ve `char[]` (char dizisi) sıkça kullanılan veri yapılarıdır. Bu iki tür, metinlerle çalışmak için farklı şekillerde kullanılan yapılardır. `string` C# dilinde metin verileri tutmak için kullanılan bir sınıftır ve bir dizi karakterden oluşur. Benzer şekilde, `char[]` ise karakterlerden oluşan bir diziyi ifade eder. Her iki tür de metinlerin işlenmesi amacıyla kullanılabilir. Bu bölümde `char[]` ve `string` veri türlerinin benzerliklerini ve farklarını inceleyeceğiz.

Şimdi size açıklaması için “copilot” a bir soru daha sordum. Hadi bir bakış atalım;

**Soru:** c programlama dilinde string değişken tipi var mı?

**Cevap:** C programlama dilinde "string" adlı özel bir veri tipi bulunmamakla birlikte, karakter dizilerini (strings) temsil etmek için karakter dizileri (character arrays) kullanılır. Bir karakter dizisi, bir char dizisidir (array of chars).

Örneğin, bir karakter dizisi tanımlamak için aşağıdaki gibi bir kod kullanabilirsiniz:

```
char myString[] = "Merhaba, Dünya!"; (c dili için geçerlidir.)
```

Bu, myString adlı bir char dizisini tanımlar ve "Merhaba, Dünya!" metnini içerir. C dilinde karakter dizileri, sonlandırıcı bir null karakter (\0) ile biter.

### `char[]` ve `string` ilişkisi

Bir `char[]` değişkeni aslında bir `string` olarak da düşünülebilir. Çünkü `char[]`, tıpkı `string` gibi birden fazla karakterden oluşur. Ancak, `char[]` veri türü dizi olduğu için dizi üzerinde yapabileceğiniz işlemlerle sınırlıdır. Örneğin, bir `char[]` dizi ile bir metni temsil edebilir ve bu metnin her bir karakterine indeks kullanarak erişebilirsiniz.

Örnek olarak aşağıdaki gibi bir `char[]` dizisi tanımlayabiliriz:

```
static void Main(string[] args)
{
    char[] harfler = { 'H', 'e', 'l', 'l', 'o' };
}
```

Şekil 8 C#'ta char[] tanımlama

Yukarıda tanımlanan `char[]` dizisi, 'H', 'e', 'l', 'l', 'o' karakterlerinden oluşur. Bu karakter dizisi, bir `string` ifadesine oldukça benzer.

`string` veri türü ise `char[]` dizisinin bir soyutlamasıdır. `string` sınıfı ya da veri tipi (sınıf kavramına nesne tabanlı programlamada değinilecektir.), metinleri işlemek için ek özellikler ve metotlar sunar. Örneğin, bir `string` ifadesinde karakter arama, değiştirme, ekleme ve benzeri birçok işlem yapmak mümkündür. `string` tanımlarken, aşağıdaki gibi kullanabilirsiniz:

```
static void Main(string[] args)
{
    string metin = "Hello";
}
```

Şekil 9 String Değişken Tanımlama

Bu örnek, "Hello" metnini `metin` değişkenine atar. Bu metin, `char[]` dizisi ile aynı karakterlerden oluşur. Ancak, `string` veri türü, bu karakterlerin ardışık olarak birleştirilmiş halini temsil eder.

Anlayacağınız üzere aslında "string" "char[]" 'e eşittir. Bu durumda bir string değişken ya da örnekteki gibi "metin" değişkeni "metin[]" olarak kullanılabilir.

## String İçerisindeki Her Bir Karaktere Erişim

Bir `string` tanımladığınızda, aslında bu `string` içerisindeki her bir karaktere dizi mantığıyla erişebilirsiniz. `string` sınıfı, karakterlere indeks kullanarak erişim sağlar. Yani, tıpkı bir `char[]` dizisinde olduğu gibi, `string` içerisindeki karakterlere de indeks numarası kullanarak ulaşabilirsiniz.

Örneğin, şekil 9'da ki `string` ifadesini ele alalım:

Bu `string` ifadesinde her bir karaktere şu şekilde erişebilirsiniz:

```
static void Main(string[] args)
{
    string metin = "Hello";
    char ilkHarf = metin[0]; // 'H'
    char ikinciHarf = metin[1]; // 'e'
    char ucuncuHarf = metin[2]; // 'l'
}
```

Şekil 10 String Değişkenin Karakterlerine Ulaşma

Yukarıdaki örnekte `metin` ifadesindeki her bir karaktere, karakterin sırasını belirten indeks numarası ile erişiyoruz. `metin[0]` ifadesi, `metin` değişkenindeki ilk karakteri (`'H'`) döndürür. Benzer şekilde `metin[1]` ifadesi, `metin` değişkenindeki ikinci karakteri (`'e'`) döndürür. Bu şekilde `string` içerisindeki her bir karaktere teker teker ulaşmak mümkündür.

Burada dikkat edilmesi gereken nokta, C# dilinde dizilerin ve `string` ifadelerin indekslerinin `0`'dan başlamasıdır. Bu yüzden `metin[0]` ifadesi, `metin` değişkenindeki ilk karakteri verir.

## `string[]` Dizisi

Bir `string` dizisi (`string[]`), birden fazla `string` ifadesini bir arada tutmak için kullanılır. `string[]`, tıpkı `int[]` veya `char[]` gibi bir dizidir ancak elemanları `string` veri tipindedir. Bu tür diziler, aynı türde birden fazla metni bir arada tutarak metinlerle çalışmayı kolaylaştırır.

Örneğin, aşağıdaki gibi bir `string[]` tanımlayabiliriz:

```
static void Main(string[] args)
{
    string[] isimler = { "Ahmet", "Ayşe", "Mehmet", "Elif" };
}
```

Şekil 11 String Dizisi

Bu `string[]` dizisi dört adet `string` ifadesinden oluşur: `"Ahmet"`, `"Ayşe"`, `"Mehmet"` ve `"Elif"`. Bu dizideki her bir `string` ifadesine, yine indeks kullanarak erişebilirsiniz:

```
static void Main(string[] args)
{
    string[] isimler = { "Ahmet", "Ayşe", "Mehmet", "Elif" };
    string birinciIsim = isimler[0]; // "Ahmet"
    string ikinciIsim = isimler[1]; // "Ayşe"
}
```

Şekil 12 String Dizisi Eleman Erişimi

Yukarıdaki örneklerde `isimler[0]` ifadesi, `isimler` dizisindeki ilk `string` ifadesini ("Ahmet") döndürürken, `isimler[1]` ifadesi ikinci `string` ifadesini ("Ayşe") döndürür.

## Jagged Array ve String Karakterlerine Erişim

`string` ve `char[]` arasındaki ilişkiyi ve `string[]` dizilerini inceledikten sonra, `jagged array` (kırık dizi, düzensiz dizi, dağınık dizi) kavramına da kısaca değinelim. `Jagged array`, elemanları farklı boyutlarda olan dizilerden oluşan bir dizidir. Yani, bir dizi içerisinde başka diziler bulundurabilirsiniz ve bu alt dizilerin boyutları birbirinden farklı olabilir. Bu kavramı isterseniz şimdilik atlayabilir daha sonra geri dönebilirsiniz. Şu anlık kafa karıştırabilir. Ancak bu konu string dizisi ve char dizisi sayesinde oldukça kolay anlaşılabilir. Bundan dolayı bu kavrama burada değinilmiştir.

Örneğin, aşağıda `jagged array` kullanarak `string` karakterlerine erişimi gösteren bir yapı tanımlayalım:

```
static void Main(string[] args)
{
    char[][] jaggedDizi = new char[3][];
    jaggedDizi[0] = new char[] { 'A', 'B', 'C' };
    jaggedDizi[1] = new char[] { 'D', 'E' };
    jaggedDizi[2] = new char[] { 'F', 'G', 'H', 'I' };
}
```

Şekil 13 jagged Array Tanımlama

Yukarıdaki örnekte, `jaggedDizi` adında bir `jagged array` tanımladık. Bu dizi, `char[]` türünde üç farklı alt dizi içerir ve her bir alt dizinin boyutu birbirinden farklıdır. Bu tür yapılar, farklı uzunluklarda dizilerle çalışırken oldukça kullanışlıdır.

Aslında şunu yaptık.

“char[]” değişken tipi art arda dizilmiş “char” değişkenlerinde oluşur. Uzunluk ya da new anahtar kelimesini anladığınızı düşünerek daha fazla detaylandırmayacağım.

“char[][]” dizisi ise ard arda dizilmiş “char[]” dizilerinden oluşur. Aslında mantık oldukça basittir.

Hatta şunu görmek hiç zor değil.



```
static void Main(string[] args)
{
    char[][][] jaggggeeeeddddDizi = new char[5][][][][];
}
```

Şekil 14 jaggggeeeeddddDizi

Şekil 14'te oldukça uç bir örnek gösterilmiştir. Gerçekte böyle bir kullanım tabi ki de yoktur. Ancak burada dizi operatör miktarını dilediğiniz gibi arttırabilirsiniz. Teoride bir kısıt yoktur. Bunu kavradığınızı düşünüyorum. Hadi biz asıl örneğimize devam edelim.

`jaggedDizi` içerisindeki belirli bir karaktere erişmek için, iki indeks kullanmanız gerekir. İlk indeks, hangi alt diziyi seçeceğinizi belirtir; ikinci indeks ise bu alt dizi içerisindeki hangi karaktere erişmek istediğinizi belirtir:

```
static void Main(string[] args)
{
    char[][] jaggedDizi = new char[3][];
    jaggedDizi[0] = new char[] { 'A', 'B', 'C' };
    jaggedDizi[1] = new char[] { 'D', 'E' };
    jaggedDizi[2] = new char[] { 'F', 'G', 'H', 'I' };

    char birinciKarakter = jaggedDizi[0][0]; // 'A'
    char ikinciKarakter = jaggedDizi[1][1]; // 'E'
    char ucuncuKarakter = jaggedDizi[2][3]; // 'I'
}
```

Şekil 15 Jagged Dizi Erişimi

Yukarıdaki örneklerde, `jaggedDizi[0][0]` ifadesi `jaggedDizi` dizisinin ilk alt dizisindeki (`{'A', 'B', 'C'}`) ilk karakteri ('A') döndürür. `jaggedDizi[1][1]` ifadesi ise ikinci alt dizideki (`{'D', 'E'}`) ikinci karakteri ('E') döndürür. Bu şekilde, `jagged array` içerisinde yer alan alt dizilerdeki karakterlere ulaşabilirsiniz.

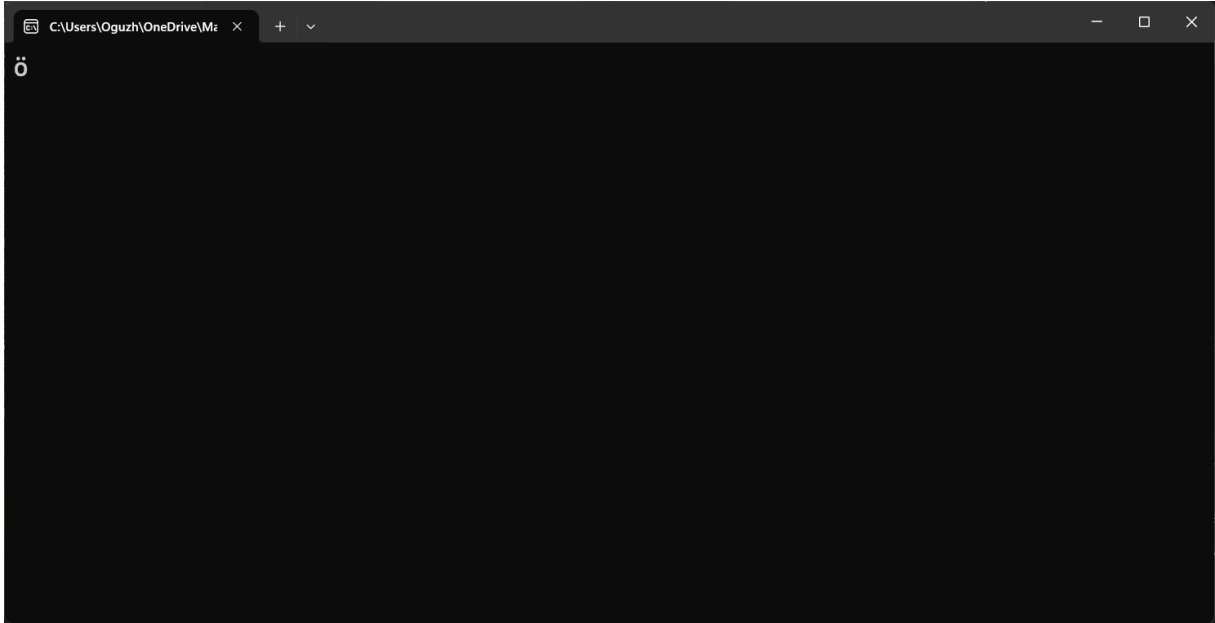
`Jagged array` yapısı, bir dizi içerisindeki alt dizilerin farklı uzunluklarda olması gerektiğinde kullanışlıdır. Bu yapıyı kullanarak, metinleri karakterlere ayırabilir ve bu karakterleri farklı alt dizilerde saklayabilirsiniz.

Şimdi biz string türünün aslında char[] olduğunu biliyoruz. Bu durumda string[] tipinin char[][] olduğu görülecektir. Hadi bunu test edelim.

```
static void Main(string[] args)
{
    string[] metinler = { "dağınık", "dizi", "örneği" };
    //doğrudan ö harfine ulaşmaya çalışalım.
    char c = metinler[2][0];
    Console.WriteLine(c);
    Console.ReadLine();
}
```

Şekil 16 String Dizisi Tek Karakter Ulaşım

Şekil 16'da bir string dizisi verilmiştir. Bu dizinin içerisinde tek bir karaktere ulaşmaya çalıştık. Bakalım programın çıktısı nasıl olacak.



Şekil 17 String Dizisi Tek Karakter Ulaşım Programı Çıktısı

Gördüğünüz üzere çıktı tamda beklediğimiz gibidir.

Metinler[2] değişkeni ile "örneği" değerine ulaştık. Metinler[2][0] ile "örneği" string ya da char[] dizisinin ilk elemanına ulaştık.

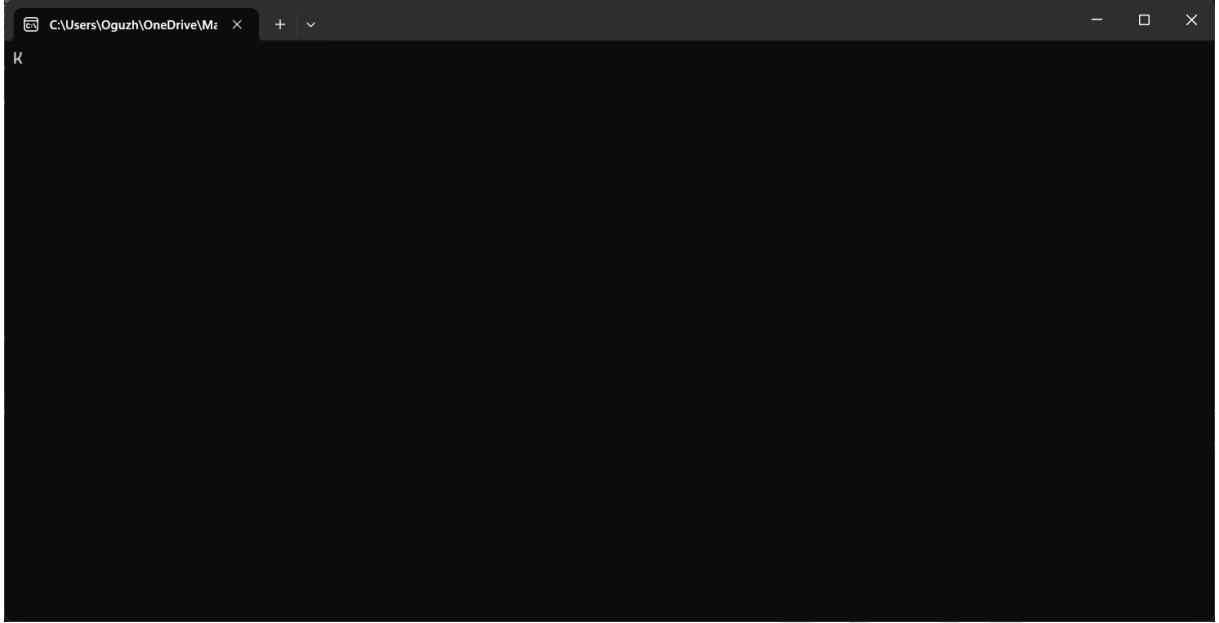
String için genel olarak şu şekilde bir yaklaşımda bulunabilirsiniz.

```
static void Main(string[] args)
{
    string soru = "işaret ile gösterilen → K ← harfini yazdır";
    Console.WriteLine(soru[24]);
    Console.ReadLine();
}
```

Şekil 18 String Değişkenine Char[] olarak Yaklaşım

İndeks 0 'dan başlayarak sayarız. Sayarken dikkat edilmesi gereken bir şey vardır; Boşluk " " bir karakterdir. Bu gibi özel karakterlere ileride değinilecektir.

Programın çıktısına bakalım.



Şekil 19 String Değişkenine Char[] olarak Yaklaşım Programı Çıktısı

Beklediğimiz üzere “K” çıktısını elde ettik. Bu programın çeşitli varyasyonlarını sizde yazarak istediğiniz karakteri ekrana yazdırabilirsiniz.

String ile ilgili bilinmesi gereken bir konu daha vardır. Bir string tanımlanırken çift tırnak "" ifadesi kullanılır. Herhangi bir metin bu çift tırnak arasına yazılır. “örnek metin” ya da “merhaba” gibi.

Kafa karışıklığına sebep olabilecek birkaç konuya burada biraz değinelim.

1. “ ” yani boşluk bir karakterdir. Anlayacağınız vardır ve karakterdir.
2. "" yani hiç ifadesi bir karakterdir. Anlayacağınız vardır ve karakterdir.

Bu iki durumu hatta özellikle 2. ifadeyi şimdilik bu şekilde bilin. Nesne tabanlı programlamada “null” ifadesini öğreneceksiniz. O zaman "" ifadesinin neden bir karakter olduğunu daha iyi anlayacaksınız.

Jagged array’leri öğrendik. Bunların düzensiz yapılar olduğunu öğrendik. Peki ya düzenli yapılar gerekiyorsa. Yani tam olarak bize [3][3][3] boyutlarında ve sürekli böyle olacak bir dizi gerekiyorsa. Hadi matrislere bir bakış atalım.

## MATRİSLER VE KULLANIMI

Matrisler, programlamada ve bilgisayar bilimlerinde önemli bir veri yapısıdır. Özellikle büyük veri kümeleri üzerinde işlem yaparken ya da verileri organize bir şekilde depolamak gerektiğinde matris yapısı kullanılır. C# programlama dilinde matrisler, çok boyutlu diziler olarak temsil edilir ve genellikle iki veya daha fazla boyuta sahip veri kümelerini yönetmek için kullanılır. Bu bölümde matrislerin nasıl tanımlandığını, jagged array (kırık diziler) ile olan farklarını, tanımlama sırasında dikkat edilmesi gereken noktaları ve ilk değer atamalarını detaylı bir şekilde ele alacağız.

## Matris Nedir?

Matris, bir dizi satır ve sütundan oluşan iki veya daha fazla boyutlu bir veri yapısıdır. Programlama dilinde matris, çok boyutlu bir dizi (multi-dimensional array) olarak temsil edilir ve genellikle satır ve sütun yapısında verileri tutmak için kullanılır. Matrisler, veri kümelerini düzenli bir şekilde saklamak ve bu veriler üzerinde çeşitli işlemler gerçekleştirmek için kullanılır. Örneğin, bir tablo veya grafikte yer alan verileri saklamak ve yönetmek için matrisler ideal bir veri yapısıdır.

Matrisler, genellikle  $m \times n$  boyutlarında ifade edilir. Burada  $m$ , satır sayısını ve  $n$ , sütun sayısını belirtir. Bir matrisin her elemanı, satır ve sütun indeksleri ile belirlenir. Örneğin,  $3 \times 3$  boyutunda bir matris, üç satır ve üç sütundan oluşan bir tablo olarak düşünülebilir.

## C# Dilinde Matris Tanımlama

C# dilinde matris tanımlamak için çok boyutlu diziler ([,] operatörü) kullanılır. Çok boyutlu diziler, birden fazla boyuta sahip verileri tutmak için kullanılır. İki boyutlu bir matris tanımlamak için aşağıdaki gibi bir tanımlama yapabiliriz:

```
static void Main(string[] args)
{
    int[,] matris = new int[3, 3];
}
```

Şekil 20 Matris Tanımlama

Yukarıdaki örnekte, `int[,]` ifadesi iki boyutlu bir `int` matrisi tanımlamak için kullanılır. `new int[3, 3]` ifadesi ise  $3 \times 3$  boyutlarında bir matris oluşturur. Bu matris, 3 satır ve 3 sütundan oluşur ve her bir elemanı `int` türündedir.

Matrisin elemanlarına değer atamak için aşağıdaki gibi bir yöntem kullanabiliriz:

```
static void Main(string[] args)
{
    int[,] matris = new int[3, 3];
    matris[0, 0] = 1;
    matris[0, 1] = 2;
    matris[0, 2] = 3;
    matris[1, 0] = 4;
    matris[1, 1] = 5;
    matris[1, 2] = 6;
    matris[2, 0] = 7;
    matris[2, 1] = 8;
    matris[2, 2] = 9;
}
```

Şekil 21 Matris Elemanlarına Değer Atama

Yukarıdaki kodda, `matris` değişkeninin her bir elemanına sırasıyla değerler atıyoruz. `matris[0, 0]` ifadesi, `matris` değişkenindeki 1. satır ve 1. sütundaki elemanı ifade eder ve bu elemanın değeri 1 olarak atanmıştır.

## Matris Elemanlarına Erişim

Bir matrisin elemanlarına erişmek için satır ve sütun indekslerini kullanırız. İki boyutlu bir dizide, ilk indeks satırı, ikinci indeks ise sütunu temsil eder. Örneğin, yukarıdaki matris değişkenindeki 2. satır ve 3. sütundaki elemana erişmek için şu şekilde bir kod kullanabiliriz:

```
static void Main(string[] args)
{
    int[,] matris = new int[3, 3];
    matris[0, 0] = 1;
    matris[0, 1] = 2;
    matris[0, 2] = 3;
    matris[1, 0] = 4;
    matris[1, 1] = 5;
    matris[1, 2] = 6;
    matris[2, 0] = 7;
    matris[2, 1] = 8;
    matris[2, 2] = 9;
    int deger = matris[1, 2]; // 6
}
```

Şekil 22 Matris Elemanlarına Erişim

Bu kod, matris değişkenindeki 2. satır ve 3. sütundaki elemanı deger değişkenine atar. Bu elemanın değeri 6'dır.

## Matris Tanımlama ve İlk Değer Ataması

Matris tanımlarken aynı zamanda elemanlarına ilk değerlerini atayabilirsiniz. C# dilinde bir matrisi tanımlarken elemanlarına ilk değerlerini atamak için süslü parantezler {} kullanılır. Aşağıda, 3 x 3 boyutlarında bir matris tanımlayıp ilk değerlerini atayan bir örnek bulunmaktadır:

```
static void Main(string[] args)
{
    int[,] matris = {
        { 1, 2, 3 },
        { 4, 5, 6 },
        { 7, 8, 9 }
    };
}
```

Şekil 23 Matris Tanımlama ve İlk Değer Ataması

Bu örnekte, matris adında 3 x 3 boyutlarında bir matris tanımlıyoruz ve her bir elemanına başlangıçta değer atıyoruz. Bu matris, aşağıdaki gibi bir yapıya sahiptir:

1 2 3

4 5 6

7 8 9

## Matris ve Jagged Array Farkları

C# dilinde matrislerle birlikte kullanılan bir diğer çok boyutlu dizi türü ise jagged array (kırık dizi) olarak adlandırılır. Jagged array, bir dizi içerisinde başka diziler bulunduran ve bu alt dizilerin boyutları birbirinden farklı olabilen bir veri yapısıdır. Yani, jagged array kullanarak her bir alt dizinin farklı boyutlarda olmasını sağlayabilirsiniz. Bu, jagged array'i klasik matrislerden ayıran önemli bir farktır.

Jagged Array bağlığında bu yapıya fazlaca değindik. Bu tür bir yapı, farklı uzunluklardaki veri kümeleriyle çalışmak gerektiğinde oldukça kullanışlıdır.

Oysa klasik bir matris (çok boyutlu dizi) tanımladığınızda, her satırda aynı sayıda sütun olmalıdır. Bu durum, veri yapısının düzenli olmasını sağlar ve bu tür matrisler üzerinde işlem yapmayı daha kolay hale getirir. Jagged array ise daha esnek bir yapı sunar ve her alt dizinin farklı boyutta olmasını mümkün kılar.

## Jagged Array ve Matris Elemanlarına Erişim

Jagged array ve matrislerin elemanlarına erişim arasında bazı farklılıklar vardır. Jagged array içerisindeki elemanlara erişmek için iki köşeli parantez ([]) kullanılır. İlk parantez, alt diziyi; ikinci parantez ise bu alt dizideki elemanı belirtir:

```
int deger = jaggedArray[1][1];
```

Bir matriste ise elemanlara erişmek için tek köşeli parantez çifti ([,]) kullanılır. Örneğin, yukarıda tanımladığımız matris değişkeninde 2. satır ve 3. sütundaki elemana şu şekilde erişebilirsiniz:

```
int deger = matris[1, 2]; // 6
```

Bu farklılık, jagged array ve matrisler arasındaki temel erişim farklarından biridir. Jagged array daha esnek bir yapı sağlarken, klasik matrisler ise daha düzenli ve sabit boyutlu bir yapı sunar.

## Matris ve Jagged Array Tanımlarken Dikkat Edilmesi Gerekenler

Matris ve jagged array tanımlarken dikkat edilmesi gereken bazı önemli noktalar vardır:

1. **Boyut Belirleme:** Bir matris veya jagged array tanımlarken boyutları doğru bir şekilde belirlemelisiniz. Çok boyutlu bir dizi kullanıyorsanız, her bir boyutun büyüklüğünü tanımlamanız gerekir. Jagged array kullanırken ise her bir alt dizinin boyutunu ayrı ayrı tanımlayabilirsiniz.
2. **Bellek Kullanımı:** Matris veya jagged array tanımlarken bellekte ayrılan alanı göz önünde bulundurmalısınız. Çok boyutlu diziler büyük miktarda bellek kullanabilir, bu nedenle özellikle büyük boyutlu diziler tanımlarken bellek kullanımı ve performans konularına dikkat etmek önemlidir.
3. **İlk Değer Atamaları:** Matris veya jagged array tanımlarken elemanlara ilk değerlerini atamak iyi bir yöntemdir. Bu sayede, dizi elemanlarına ilk değer verilmemesi durumunda oluşabilecek belirsizliklerin önüne geçilir. Örneğin, int türünde bir matris tanımladığınızda, elemanların

varsayılan değeri 0 olacaktır. Ancak, her elemanı belirli bir değere atamak, kodun daha anlaşılır ve güvenli olmasını sağlar.

4. **Erişim Yöntemleri:** Matris ve jagged array elemanlarına erişirken doğru indeksleri kullandığınızdan emin olmalısınız. Yanlış bir indeks kullanımı, `IndexOutOfRangeException` gibi hatalara yol açabilir. Bu yüzden indekslerin geçerli aralıkta olup olmadığını kontrol etmek önemlidir.
5. **Kullanım Amacı:** Jagged array ve matris arasındaki seçim, kullanmak istediğiniz amaca göre yapılmalıdır. Eğer her satırda farklı sayıda eleman tutmak istiyorsanız, jagged array kullanmak daha uygun olacaktır. Ancak, her satırda aynı sayıda sütun olmasını istiyorsanız ve verilerin düzenli bir yapıda saklanması gerekiyorsa, klasik matris (çok boyutlu dizi) kullanmak daha mantıklıdır.

## Matris ve Jagged Array Örnekleri

Aşağıda matris ve jagged array kullanarak veri saklama ve bu verilere erişimle ilgili çeşitli örnekler bulacaksınız. Bu örnekler, matrislerin ve jagged arraylerin nasıl tanımlandığını, kullanıldığını ve elemanlarına nasıl erişildiğini göstermektedir.

### Matris Örneği

Matrisler, genellikle düzenli ve sabit boyutlu veri kümelerini saklamak için kullanılır. Örneğin, bir sınıftaki öğrencilerin farklı derslerden aldığı notları tutmak için bir matris kullanılabilir.

```
int[,] notlar = {  
    { 85, 90, 78 },  
    { 88, 92, 80 },  
    { 75, 85, 82 }  
};
```

Yukarıdaki örnekte, notlar adında bir 3 x 3 boyutunda matris tanımlanmıştır. Bu matris, üç öğrencinin üç farklı dersten aldığı notları temsil etmektedir. Her satır bir öğrenciyi, her sütun ise bir dersti temsil eder.

Matris elemanlarına erişmek için satır ve sütun indekslerini kullanabiliriz:

```
int ogrenci1_ders1_notu = notlar[0, 0]; // 85  
int ogrenci2_ders3_notu = notlar[1, 2]; // 80
```

Yukarıdaki kodda, `notlar[0, 0]` ifadesi birinci öğrencinin birinci dersten aldığı notu (85) döndürür. `notlar[1, 2]` ifadesi ise ikinci öğrencinin üçüncü dersten aldığı notu (80) döndürür.

## Jagged Array Örneği

Jagged array (kırık dizi), her bir alt dizinin farklı boyutlarda olmasını sağlayan esnek bir yapıdır. Örneğin, bir okuldaki farklı sınıflarda bulunan öğrenci sayıları farklı olabilir ve her sınıftaki öğrencilerin notlarını tutmak için jagged array kullanılabilir.

```
int[][] sinifNotlari = new int[3][];
```

```
sinifNotlari[0] = new int[] { 85, 90, 78 };
```

```
sinifNotlari[1] = new int[] { 88, 92 };
```

```
sinifNotlari[2] = new int[] { 75, 85, 82, 91 };
```

Yukarıdaki örnekte, sinifNotlari adında bir jagged array tanımlanmıştır. Bu jagged array, üç farklı sınıftaki öğrencilerin notlarını tutmaktadır. Her alt dizinin boyutu farklıdır, bu da her sınıftaki öğrenci sayısının farklı olduğunu gösterir.

Jagged array elemanlarına erişmek için iki indeks kullanılır:

```
int sinif1_ogrenci1_notu = sinifNotlari[0][0]; // 85
```

```
int sinif3_ogrenci4_notu = sinifNotlari[2][3]; // 91
```

Yukarıdaki kodda, sinifNotlari[0][0] ifadesi birinci sınıftaki birinci öğrencinin notunu (85) döndürür. sinifNotlari[2][3] ifadesi ise üçüncü sınıftaki dördüncü öğrencinin notunu (91) döndürür.

### Matris ve Jagged Array Kullanım Senaryoları

Matrisler ve jagged arrayler farklı kullanım senaryolarında tercih edilirler. Bu farkları anlamak, doğru veri yapısını seçmek açısından önemlidir:

- **Matrisler:** Eğer her satırda aynı sayıda sütun olması gerekiyorsa ve veriler düzenli bir yapıdaysa, matris kullanmak daha uygundur. Örneğin, bir çalışan tablosunda her çalışanın maaşı, departmanı ve yaşı gibi bilgiler düzenli bir şekilde tutulabilir.
- **Jagged Array:** Eğer her satırda farklı sayıda eleman olması gerekiyorsa ve veri kümeleri esnek boyutlara sahipse, jagged array kullanmak daha mantıklıdır. Örneğin, farklı sayıda öğrencisi olan sınıfların notlarını tutmak için jagged array kullanmak daha uygundur.

Tıpkı char[][][] jagged array örneği gibi matrisinizinde boyutunu [,,,] şeklinde artırabilirsiniz. Kısa bir örnek verelim.



```
int[,,,] matris5D = {
    {
        { { 1, 2 }, { 3, 4 } },
        { { 5, 6 }, { 7, 8 } }
    },
    {
        { { 9, 10 }, { 11, 12 } },
        { { 13, 14 }, { 15, 16 } }
    },
    {
        { { 17, 18 }, { 19, 20 } },
        { { 21, 22 }, { 23, 24 } }
    },
    {
        { { 25, 26 }, { 27, 28 } },
        { { 29, 30 }, { 31, 32 } }
    }
};
```

Şekil 24 5 Boyutlu Matris

Tabiki de bu da çok uç bir örnektir. Sadece görmemiz için oluşturdum. Kafanız karışmasın böyle bir şeyi hiç kullanmayacağız.

Ancak buradaki elemanlarada ulaşmak aslında çok kolaydır.

```
int deger = matris5D[1, 1, 1, 0, 1]; // 27
```

Bu konuları şimdilik sadece bilmeniz yeterlidir. Uygulamalarımızda bu şekilde kullanımda bulunmayacağız. Bu notların burada olduğunu ve yeri geldiğinde dönüp bu notları okuyup bilgi alabileceğinizi unutmayın.

## DİZİ ELEMANLARINA ADRESLERİ İLE DOĞRUDAN ULAŞMA ÖRNEĞİ

Bir önceki derste şöyle bir örnek yapmıştık. İnt değişken tipi 4 byte ve “sayı” adlı değişken en anlamsız byte’ın adresini temsil ediyor. Bu adrese 1 ekleyerek diğer byte’lara da ulaşmıştık. Şimdi diziler içinde bu örneği tekrarlayalım.

Kolay olması için bir byte array kullanalım. Eğer merakınız varsa sizler bunu int array ile yapabilirsiniz. Chatgpt’den yardım alarak kesinlikle başarabilirsiniz.

Bir byte array oluşturalım. Her bir eleman 1 byte olacaktır. Değişkenimizin konumu ise ilk elemanın adresi olacaktır. Biz bu adrese array’in boyutu kadar 1 ekleyerek her bir elemana adresler ile doğrudan ulaşabiliriz.

“char” veri tipinin aslında sayı olduğunu da biliyoruz. Ascii tablosundan bakarak sayıların hangi harflere karşılık geldiğini bulun ve istediğiniz kelimenin harflerini teker teker byte array’e yazın.

Ben burada hello, world kullandım. Siz istediğinizi yazabilirsiniz.

Hadi deneyelim.

```
unsafe static void Main(string[] args)
{
    // Bir örnek array tanımlayalım
    byte[] data = { 72, 101, 108, 108, 111, 44, 32, 87, 111, 114, 108, 100, 33 }; // "Hello, World!"

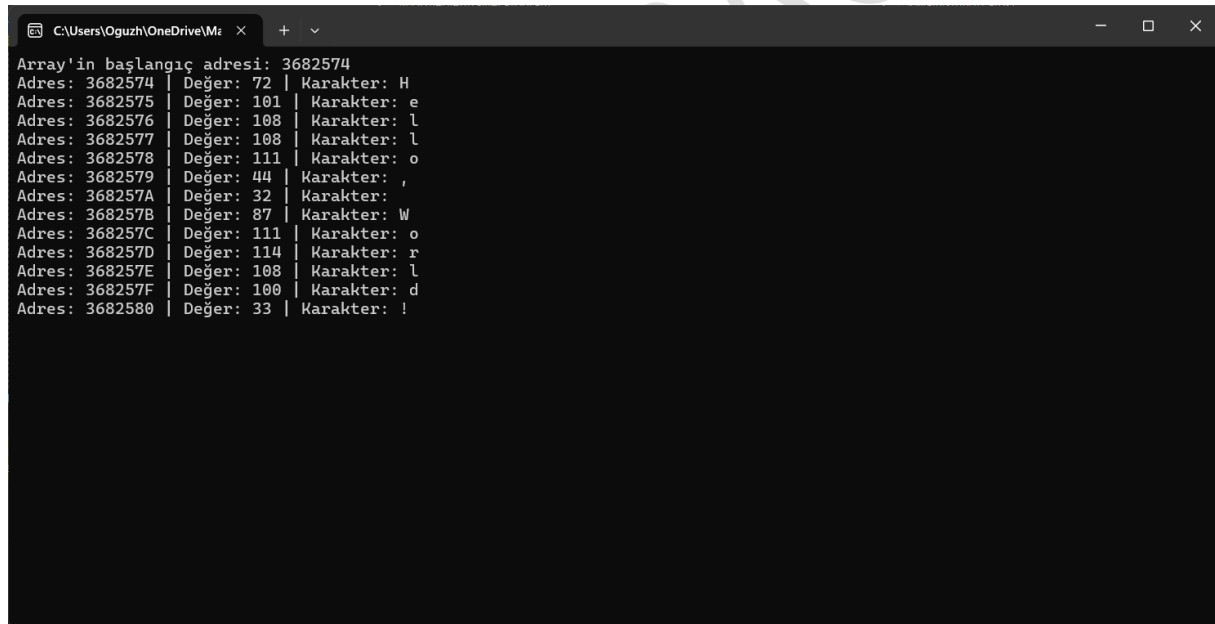
    // Array'in başlangıç adresini sabitlemek için fixed kullanıyoruz
    fixed (byte* ptr = data)
    {
        Console.WriteLine("Array'in başlangıç adresi: {0:X}", (long)ptr);

        // Array'in her elemanının adresini ve değerini yazdıralım
        for (int i = 0; i < data.Length; i++)
        {
            byte* currentAddress = ptr + i; // Elemanın gerçek adresi
            Console.WriteLine($"Adres: {(long)currentAddress:X} | Değer: {*(currentAddress)} | Karakter: {(char)*
                (currentAddress)}");
        }
    }
    Console.ReadLine();
}
```

Şekil 25 Byte Array Adres İle Elemanlara Teker Teker Ulaşma

Burada hello,world kelimesinin ascii karşılığını bir byte array'e verdik. Ardından adresi, sayıyı ve sayının karakter karşılığını yazdırdık.

Bakalım sonuç beklediğimiz gibi mi?



```
Array'in başlangıç adresi: 3682574
Adres: 3682574 | Değer: 72 | Karakter: H
Adres: 3682575 | Değer: 101 | Karakter: e
Adres: 3682576 | Değer: 108 | Karakter: l
Adres: 3682577 | Değer: 108 | Karakter: l
Adres: 3682578 | Değer: 111 | Karakter: o
Adres: 3682579 | Değer: 44 | Karakter: ,
Adres: 368257A | Değer: 32 | Karakter: 
Adres: 368257B | Değer: 87 | Karakter: W
Adres: 368257C | Değer: 111 | Karakter: o
Adres: 368257D | Değer: 114 | Karakter: r
Adres: 368257E | Değer: 108 | Karakter: l
Adres: 368257F | Değer: 100 | Karakter: d
Adres: 3682580 | Değer: 33 | Karakter: !
```

Şekil 26 Byte Array Adres İle Elemanlara Teker Teker Ulaşma Program Çıktısı

Tam olarak beklediğimiz gibi. Eğer programı anlamadıysanız chatgpt ile konuşun. Şu an for döngüsünü bilmediğinizden bu program size kafa karıştırıcı gelecektir. Ancak yine de programın temel mantığını anlamanızda bir engel yoktur.

## ÖZET VE SONUÇ

C# dilinde diziler, aynı türdeki verileri bir arada tutan veri yapılarıdır. Diziler [] operatörü kullanılarak tanımlanır ve new anahtar kelimesiyle oluşturulur. İlk değer ataması {} parantezleri içinde yapılabilir. Örneğin, `int[] sayilar = new int[5];` veya `int[] sayilar = {1, 2, 3, 4, 5};` şeklinde tanımlama yapılabilir.

Dizilere eleman eklemek için indeksler ([i]) kullanılır. Dizilerin boyutu sabittir ve bellekte bu boyut kadar yer ayrılır. Jagged diziler ise farklı boyutlarda alt diziler içerebilir ve bu esneklik, farklı veri kümelerini saklamak için kullanışlıdır.

## ÖDEV

- Bir sonraki ders için string kaçış elemanlarını araştırınız. Bunun hakkında kendi dokümanınızı oluşturunuz.
- Her tipte verilerden dizi oluşturunuz. İlk değerlerini atayınız ve rastgele bir elemana ulaşmaya çalışınız.
- En az iki jagged array ve matris oluşturun. Bunlar en fazla 3 boyutlu olsun ve daha fazlası ile uğraşmayın. İlk değerlerini atamaya çalışın. Bu çalışmada ekrana yazdırmak o kadar önemli değildir. Önemli olan hata almadan ilk değer atamasını yapabilmektir. Bu yetenek ileride çok işinize yarayacaktır. Yapay zekadan yardım alabilirsiniz. Ancak en az bir kere yardımsız oluşturunuz.

## TEST

Her bir soru 5 puandır. Eğer 60'ın altında alırsanız lütfen konuyu tekrar ediniz. Kopya çekmek serbest. Kopyala yapıştır yasak.

1. **C# dilinde diziler nedir?**
  - a) Birkaç farklı veri tipini bir arada tutan yapı
  - b) Aynı veri tipindeki birden fazla değeri bir arada tutan yapı
  - c) Herhangi bir veri tipini saklayamayan yapı
  - d) Sadece string verileri saklayan yapı
2. **Bir dizi tanımlarken [] operatörünün görevi nedir?**
  - a) Dizin türünü belirler
  - b) Dizin boyutunu belirler
  - c) Diziye erişimi sağlar
  - d) Dizi siler
3. **Aşağıdaki ifadelerden hangisi doğru bir dizi tanımlamasıdır?**
  - a) `int[] dizi = [5];`
  - b) `int[] dizi = new int[5];`
  - c) `int[5] dizi = new int[];`
  - d) `int[] dizi = new[5];`
4. **C# dilinde dizilerde indeks numaraları kaçtan başlar?**
  - a) 0
  - b) 1
  - c) -1
  - d) Dizin türüne bağlıdır

5. **new** anahtar kelimesi C# dilinde ne için kullanılır?
- a) Değişkeni silmek
  - b) Değişken oluşturmak ve ilk değer atamak
  - c) Dizin boyutunu artırmak
  - d) Diziyi kopyalamak
6. **char[]** ve **string** arasındaki ilişki nedir?
- a) **char[]** bir **string** değildir
  - b) **string** bir **char[]** dizisinin soyutlamasıdır
  - c) **string** bir **int[]** dizisinin türevidir
  - d) **char[]** ve **string** aynı veri tipidir
7. **string[]** dizisinin her bir elemanı hangi türdendir?
- a) **int**
  - b) **char**
  - c) **string**
  - d) **bool**
8. **jagged array** nedir?
- a) Aynı boyuttaki dizilerin birleşimi
  - b) Farklı boyutlarda alt diziler içeren bir dizi
  - c) Sadece tek boyutlu dizilerden oluşan yapı
  - d) Aynı türdeki elemanları tek bir satırda tutan yapı
9. **Matris** ve **jagged array** arasındaki fark nedir?
- a) Matrislerin boyutu değişken olabilir
  - b) Jagged array esnek boyutlu alt diziler içerir
  - c) Matrisler her zaman tek boyutludur
  - d) Jagged array sadece **string** verileri saklar
10. Dizi elemanlarına erişirken hangi operatör kullanılır?
- a) {}
  - b) ()
  - c) []
  - d) <>
11. Aşağıdaki kodda **notlar[2, 1]** hangi değeri döndürür?
- ```
int[, ] notlar = { { 85, 90, 78 }, { 88, 92, 80 }, { 75, 85, 82 } };
```
- a) 90
  - b) 85
  - c) 82
  - d) 80
12. **new int[5]** ifadesi ne anlama gelir?
- a) 5 elemanlı bir dizi oluşturur
  - b) Diziyi siler
  - c) Dizi elemanlarını 5 ile doldurur
  - d) 5 boyutlu bir dizi oluşturur

13. `matris[1, 2]` ifadesi neyi temsil eder?
- a) Matrisin ilk elemanını
  - b) 1. satır ve 2. sütundaki elemanı
  - c) 2. satır ve 3. sütundaki elemanı
  - d) Matrisin son elemanını
14. jagged array tanımlarken her alt dizinin boyutu nasıl belirlenir?
- a) Önceden belirlenir ve değiştirilemez
  - b) Her alt diziye ayrı ayrı belirlenir
  - c) Sadece bir alt dizi belirlenir
  - d) Otomatik olarak atanır
15. Dizilerin varsayılan değerleri nelerdir? (`int[]` için)
- a) 1
  - b) 0
  - c) null
  - d) -1
16. **C# dilinde** string bir sınıf mıdır?
- a) Evet
  - b) Hayır
  - c) Sadece bazen
  - d) Belirtilmemiştir
17. Matrislerde elemanlara nasıl erişilir?
- a) `()` kullanılarak
  - b) `[,]` kullanılarak
  - c) `{}` kullanılarak
  - d) `<>` kullanılarak
18. Aşağıdakilerden hangisi jagged array tanımlamak için doğru bir ifadedir?
- a) `int[][] dizi = new int[3][];`
  - b) `int[,] dizi = new int[3,3];`
  - c) `int dizi = new int[3];`
  - d) `int[] dizi = new jagged[3];`
19. Bir `char[]` dizisi tanımlandığında bu dizinin elemanlarına nasıl erişilir?
- a) `[]` operatörü kullanarak
  - b) `()` operatörü kullanarak
  - c) `{}` operatörü kullanarak
  - d) `<>` operatörü kullanarak
20. Matris tanımlarken `int[,]` ifadesi ne anlama gelir?
- a) Tek boyutlu dizi
  - b) İki boyutlu matris
  - c) Üç boyutlu matris
  - d) Beş boyutlu matris

## CEVAP ANAHTARI

1. b
2. c
3. b
4. a
5. b
6. b
7. c
8. b
9. b
10. c
11. b
12. a
13. c
14. b
15. b
16. a
17. b
18. a
19. a
20. c