# BÖLÜM 9: DİZİLER

Anahtar Kavramlar
Tek Boyutlu Diziler
İndis
Karakter Dizileri
Sonlandırma Karakteri
Seriler
Arama
Sıralama
Çok Boyutlu Diziler
Matris

## Öğrenme Hedefleri

1. Dizileri nasıl tanımlayacağını ve dizilerin elemanlarına nasıl erişim sağlanabileceğini bilir.
2. Tanımlı dizilerin elemanlarına atanmış değerlerin nasıl güncellenebileceğini bilir.
3. Dizilerin fonksiyonlara nasıl aktarılabileceğini bilir.
4. Tek boyutlu dizlerin yanı sıra çok boyutlu dizilerin de nasıl tanımlanacağını ve çok boyutlu dizilerin elemanlarına nasıl erişim sağlanabileceğini bilir.
5. Dizileri kullanarak arama ve sıralama gibi belirli problemlerin çözümü için algoritmalar geliştirebilir.
6. Diziler üzerinde matematiksel işlemler gerçekleştirebilir.

# 1 GİRİŞ

C++ programlama dilinde diziler, aynı veri türünden belirli sayıda elemanı bellekte sıralı bir şekilde saklayan yapılardır. Aynı veri türünden çok sayıda elamanın işlendiği durum için diziler, diğer yapılara nazaran daha etkili ve pratik araçlar olarak öne çıkmaktadır. Dizilerde saklanacak elemanların sonlu sayıda ve aynı veri türünde olması gerekmektedir. Dizilerdeki elemanlara erişim ilgili elemanların bellekteki sırasını belirten indisler yoluyla yapılmaktadır. Bu bölümde C++ programlama dilinde tanımlanmış tek boyutlu ve iki boyutlu dizilerden bahsedilecektir.

## Tek Boyutlu Diziler

Matematikteki vektör kavramının C++ programlama dilinde karşılığı olarak değerlendirilebilecek diziler aşağıdaki kod parçası üzerinden tanımlanmaktadır:

```
veri_turu diziAdi[boyut];
```

Burada *veri_turu* bellekte saklanacak verilerin türünü, *boyut* da verilerin sayısını belirtmektedir. Dizilerin bellekte kapladığı toplam alan dizinin boyutunun yanı sıra veri türü ile de doğrudan ilintilidir. Örneğin aynı boyuta sahip kesirli sayılar için oluşturulmuş bir dizi tam sayılar için oluşturulmuş bir diziye göre bellekte daha çok alan kaplayacaktır.

```
int dizi[4];
```

Örneğin yukarıdaki kod parçası ile 4 elemanlı bir tam sayı dizisi tanımlanmaktadır. Giriş kısmında da belirtildiği gibi tanımlanan dizinin elemanlarına, elemanların indislerini alt simge işleci [ ] içerisine yazarak erişim sağlanmaktadır. Her bir elemanın indisi dizideki sıralarının bir eksiği olarak belirlenmektedir. Örneğin yukarıda tanımlanan tam sayı dizisinde dizinin üçüncü elemanına erişim sağlamak için dizi[2] kullanılmalıdır. Yine aynı sayı dizisinin elemanları sırasıyla dizi[0], dizi[1], dizi[2] ve dizi[3] olacaktır.

Dizinin elemanlarına değer atama işlemi dizi tanımlandığında yapılabileceği gibi, dizi tanımlandıktan sonra ihtiyaca göre kullanıcıdan gelen girdiler veya program içerisinde uygulanan işlemler üzerinden de gerçekleştirilebilmektedir. C++ programlama dilinde bir dizinin elemanlarına tanımlama esnasında değer atama işlemi küme parantezi yoluyla yapılabilmektedir. Örneğin aşağıdaki kod parçasında 4 elemanlı bir tam sayı dizisi tanımlanmakta ve ilgili dizinin elemanlarına sırasıyla 1, 2, 3, 4 değerleri atanmaktadır.

```
int dizi[4] = {1, 2, 3, 4};
```

Küme parantezi yardımıyla atama işleminde dizinin boyutundan daha az sayıda değer girilmişse, program kalan elemanlara tam sayılarda tanımlanmış diziler için 0 değerini atayacaktır. Örneğin aşağıdaki kod bloğunda, program ilk tam sayı dizisinin ilk iki elemanına sırasıyla 1 ve 2 değerlerini atarken, son iki elemana 0 değerini atayacak; ikinci tam sayı dizisi için küme parantezi içerisine bir değer girilmediğinden, program ikinci sayı dizisinin bütün elemanlarına 0 değerini atayacaktır. Diğer bir ifadeyle ilk sayı dizisinin elemanlarının aldığı değerler sırasıyla *dizi_1[0] = 1, dizi_1[1] = 2, dizi_1[2] = 0 ve dizi_1[3] = 0* iken, ikinci sayı dizisinin elemanlarının aldığı değerler sırasıyla *dizi_2[0] = 0, dizi_2[1] = 0, dizi_2[2] = 0 ve dizi_2[3] = 0 tür.*

```
int dizi_1[4] = {1, 2};
int dizi_2[4] = {};
```

**DİKKAT EDELİM**

Küme parantezi yardımıyla atama işleminde dizinin boyutundan daha az sayıda değer girilmişse, program kalan elemanlara tam sayılarda tanımlanmış diziler için 0 değerini atayacaktır.

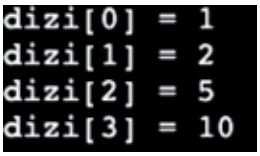
Diğer yandan tanımlanan bir dizinin elemanlarına sonradan değer atamak ya da atanmış değerleri ihtiyaca göre değiştirmek aşağıdaki kod bloğunda gösterildiği gibi yine atama işleci yardımıyla yapılmaktadır. Kod bloğunda 4 elemanlı bir tam sayı dizisi tanımlanmış ve atama işleci yardımıyla dizinin ilk iki elemanına sırasıyla 1 ve 2 değerleri atanmıştır. Sonrasında 'dizi[1] = dizi[0] + dizi[1]' kod satırı ile dizinin ikinci elemanının değeri değiştirilmiştir.

```
int dizi[4];
dizi[0] = 1, dizi[1] = 2;
dizi[1] = dizi[0] + dizi[1];
```

Tanımlanmış dizilerin tüm elemanlarına değer atamak, atanmış değerleri değiştirmek ya da tüm elemanlara erişim sağlamak gibi işlemler for döngüsü yoluyla yapılabilir.

```
int dizi[4];
int indis;
for (indis = 0; indis < 4; indis++){
        dizi[indis] = indis*indis;
}
for (indis = 0; indis < 4; indis++){
        dizi[indis] = dizi[indis] + 1;
}
for (indis = 0; indis < 4; indis++){
        cout << "dizi[" << indis << "] = "<< dizi[indis] << endl;
}
```

Yukarıdaki kod bloğunda 4 elemanlı bir sayı dizisi tanımlanmış ve ilk for döngüsünde 'dizi[indis] = indis*indis' kod satırı üzerinden dizinin her bir elemanına, ilgili elemanın indisinin karesi değer olarak atanmıştır. Yani ilk for döngüsü çalıştırıldıktan sonra dizinin elemanlarının alacağı değerler sırasıyla *dizi[0] = 0, dizi[1] = 1, dizi[2] = 4 ve dizi[3] = 9* olacaktır. Kod bloğundaki ikinci for döngüsünde ise 'dizi[indis] = dizi[indis] + 1' kod satırı üzerinden dizinin her bir elemanının değeri bir fazlası olarak yenilenmektedir. Kod bloğundaki son for döngüsünde ise dizinin her bir elemanının güncel değeri 'cout << "dizi[" << indis << "] = "<< dizi[indis] << endl' kod satırı yoluyla ekrana yazdırılmaktadır. Dolayısıyla kod bloğunun ekran çıktısı Şekil 9.1'deki gibi olacaktır.




*Şekil 9.1 Tam Sayı Dizisinin Elemanlarının Ekrana Yazdırılması.*

Hatırlanacağı üzere matematikteki seriler dizilerin belirli bir kesitindeki sayıların (ya da bütün sayıların) toplamı olarak ifade edilmektedir. Serilerin C++ programlama dilindeki karşılığı dizilere uygulanacak for döngüsü yardımıyla elde edilebilir. Örneğin aşağıdaki kod bloğunda 6 elemanlı bir tam sayı dizisi tanımlanmış ve bu dizinin elemanlarına ilk for döngüsündeki 'dizi[indis] = 10*indis + 5' kod satırı üzerinden, başlangıç değeri 5 ve ortak farkı 10 olan aritmetik dizinin ilk 6 elemanı değer olarak atanmıştır. Dolayısıyla ilgili dizinin elemanları *sırasıyla dizi[0] = 5, dizi[1] = 15, dizi[2] = 25, dizi[3] = 35, dizi[4] = 45 ve dizi[5] = 55* olacaktır. Kod bloğundaki ikinci for döngüsündeki 'toplam += dizi[indis]' kod satırı yoluyla ilgili dizinin terimleri toplanacaktır. Kod bloğu çalıştırıldığında toplam değişkeninin alacağı değer 180 olacaktır. Kod bloğundaki son for döngüsünde ise 'ara_toplam += dizi[indis]' kod satırı üzerinden dizinin ikinci elemanından beşinci elemanına kadar olan elemanların toplamı hesaplanacak ve bu toplam *ara_toplam* değişkeninde tutulacaktır. Kod bloğu çalıştırıldığında *ara_toplam* değişkeninin alacağı değer 120 olacaktır.

```
int dizi[6];
int indis;
int toplam = 0;
int ara_toplam = 0;
for (indis = 0; indis < 6; indis++){
        dizi[indis] = 10*indis + 5;
}
for (indis = 0; indis < 6; indis++){
        toplam + = dizi[indis];
}
for (indis = 1; indis < 5; indis++){
        ara_toplam + = dizi[indis];
}
```

### Karakter Dizileri

Yukarıda da belirtildiği gibi, karakter dizileri de tam sayı dizilerine benzer aşağıdaki kod satırı üzerinden tanımlanmaktadır.

```
char diziAdi[boyut];
```

Tanımlamış karakter dizisinin elemanlarına değer atama işlemi, tam sayılarda olduğu küme parantezi yardımıyla yapılabileceği gibi atanacak değerlerin çift tırnak içinde tek bir kelime olarak yazılması ile de mümkündür. Yine tam sayılardakine benzer olarak, karakter dizilerinde ilk değer atama işleminde dizinin boyutundan daha az sayıda değer girilmişse, program kalan elemanlar için boş karakteri değer olarak atayacaktır.

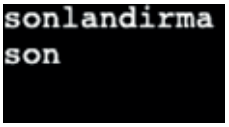
```
char dizi_1[4] = {'a', 'd', 'a'};
char dizi_2[4] = {};
char dizi_3[4] = "ad";
```

Örneğin yukarıdaki kod parçasında 4 uzunluklu 3 farklı karakter dizisi tanımlanmış ve küme parantezi yardımıyla ilk dizinin ilk üç elemanına sırasıyla 'a', 'd' ve 'a' değerleri atanmıştır. Dizinin son elemanı için bir değer belirtilmediğinden program son elemana boş karakter değerini atayacaktır. Dolayısıyla ilk dizinin elemanları sırasıyla *dizi_1[0] = 'a', dizi_1[1] = 'd', dizi_1[2] = 'a' ve dizi_1[3] = ' '* olacaktır. Küme parantezi içerisinde bir karakter girilmediğinden program ikinci dizinin bütün elemanlarına boş karakter değerini atayacaktır. Dolayısıyla ikinci dizinin elemanları sırasıyla *dizi_2[0] = ' ', dizi_2[1] = ' ', dizi_2[2] = ' ' ve dizi_2[3] = ' '* olacaktır. Üçüncü dizide ise değer atama işlemi çift tırnak içerisine yazılan karakterler yoluyla yapılmaktadır. Çift tırnak içerisine yazılan karakterlerin sayısı dizinin boyutundan az olduğu için program dizinin kalan elemanlarına boş karakter değerini atayacaktır. Çift tırnak değer atama yönteminde program, küme parantezinden farklı olarak dizinin değer girilmemiş kalan elemanlarından ilkine sonlandırma karakteri '\0' değerini atayacaktır. Sadece karakter dizileri için tanımlanmış sonlandırma karakteri '\0'; bellekte ayrılmış alanın nereye kadar kullanıldığını, bellekteki ilgili alanın tümünü kontrol etmeksizin tespit edebilmemize imkan sağlamaktadır. Son dizinin elemanları değerler atandıktan sonra sırasıyla *dizi_3[0] = 'a', dizi_3[1] = 'd', dizi_3[2] = '\0 ' ve dizi_3[3] = ' '* olacaktır. Diğer yandan dizinin elemanlarını ekrana yazdırmak istediğimizde program sonlandırma karakterini pas geçecektir.

Sonlandırma karakteri, yukarıda belirtildiği gibi çift tırnak yöntemi üzerinden atanabileceği gibi, aşağı kod bloğunda olduğu gibi küme parantezi yöntemi üzerinden de değer olarak atanabilmektedir. Aşağıdaki kod bloğunda 15 uzunluklu bir karakter dizisi tanımlanmış ve dizinin elemanlarına küme parantezi yöntemi yardımıyla sırasıyla 's', 'o', 'n', 'l', 'a', 'n', 'd', 'i', 'r', 'm', 'a', ve '\0' değerleri atanmıştır. Sonrasında 'dizi[3] = '\0' ' kod satırı üzerinden dizinin dördüncü elemanı sonladırma

karakteri olarak yenilenmiştir. Kod parçası çalıştırıldığında alınacak ekran çıktısı Şekil 9.2'de verilmiştir. Şekil 9.2'den de anlaşılacağı gibi dizinin elemanları tekrar ekrana yazdırılmak istendiğinde program sadece '\0' sonlandırma karakterine kadar olan karakterleri ekrana yazdırmıştır.

```
char dizi[15] = {'s', 'o', 'n', 'l', 'a', 'n', 'd', 'i', 'r', 'm', 'a', '\0' };
cout << dizi << endl;
dizi[3] = '\0';
cout << dizi << endl;
```




*Şekil 9.2 Karakter Dizisinin Elemanlarının Ekrana Yazdırılması.*

**DİKKAT EDELİM**

Sadece karakter dizileri için tanımlanmış sonlandırma karakteri '\0'; bellekte ayrılmış alanın nereye kadar kullanıldığını, bellekteki ilgili alanın tümünü kontrol etmeksizin tespit edebilmemize imkan sağlamaktadır.

### Dizilerde Arama

Aşağıdaki kod bloğu tanımlı bir dizinin kullanıcı tarafından girilen bir elemanı içerip içermediğini belirlemektedir. Kod bloğunda önce 15 uzunluklu bir tamsayı dizisi tanımlanmış ve bu dizinin elemanlarına sırasıyla 7, 12, 6, 21, 17, 5, 12, 32, 9, 2, 16, 31, 3, 18, 23 değerleri atanmıştır. Sonrasında kullanıcıdan pozitif bir tamsayı değeri girmesi istenmiş, while döngüsü ve indis tam sayı değişkeni yardımıyla dizinin girilen tam sayıyı içerip içermediği belirlenmiştir. Dikkat edilirse kod bloğunda dizinin boyutu '(sizeof(diz)/sizeof(*diz))' formülü yardımıyla tanımlanmaktadır. Örneğin kullanıcı 17 değerini girerse; while döngüsünün beşinci adımında if kontrol yapısındaki 'diz[indis] ==x' koşulu doğru olacağından, program kontrol yapısının gövdesindeki kodu break anahtar kelimesine kadar çalıştıracak ve ekrana "Girilen deger dizinin 5. elemanidir" cümlesini yazdıracaktır. Diğer yandan kullanıcı 1 değerini girerse; herhangi bir indis değeri için döngü içerisindeki 'diz[indis] ==x' koşulu doğru olamayacak ve döngü sonunda indis tamsayı değişkeninin değeri 15 olacaktır. Dolayısıyla while döngüsü dışındaki if kontrol yapısının koşulu 'indis >= (sizeof(diz)/sizeof(*diz))' doğru olacak ve program kontrol yapısının gövdesindeki kodu çalıştırarak ekrana "Dizi, girilen tamsayi değerini icermemektedir" cümlesini yazdıracaktır.

```
int diz[15] = {7, 12, 6, 21, 17, 5, 12, 32, 9, 2, 16, 31, 3, 18, 23};
int indis = 0, x;
cout << "Pozitif bir tamsayi degeri giriniz : " << endl;
cin >> x;
```

```
    int boyut = (sizeof(diz)/sizeof(*diz));
    while (indis < boyut){
        if (diz[indis] == x){
            cout << "Girilen deger dizinin " << indis +
1 << ". elemanidir" << endl;
            break;
        }
        indis++;
    }
    if(indis >= boyut)
        cout << "Dizi, girilen tamsayi degerini icer-
memektedir" << endl;
```

### Tek Boyutlu Dizilerin Fonksiyonlara Aktarılması

Tanımlı bir dizinin girilen bir elemanı içerip içermediğini belirleyen yukarıdaki program aşağıda gösterildiği gibi bir fonksiyon yardımıyla da tanımlanabilir. Aşağıdaki tanımı verilen Arama fonksiyonu bir diziyi, dizinin boyutunu ve dizide aranacak x elemanını girdi olarak almakta; ve for döngüsü yardımıyla dizi x elemanını içeriyorsa ilgili elemanın dizideki sırasına, içermiyorsa 0 değerine dönmektedir.

```
int Arama(int dizi[], int boyut, int x) {
    for(int i = 0; i < boyut;i++){
        if(dizi[i] ==x)
            return (i+1);



    }
    return 0;
}
```

Aşağıda verilen kod bloğu ise yukarıda tanımlanan Arama fonksiyonu yardımıyla tanımlı bir dizinin girilen bir tamsayı değerini içerip içermediğini belirlemektedir. Örneğin kullanıcı *5* değerini girerse; *Arama(diz, boyut, x)* fonksiyonu verilen girdi değerleri için 6 değerine döneceğinden, 'Arama(diz, boyut, x) == 0' koşulu yanlış olacak ve program if..else kontrol yapısındaki alternatif kodu çalıştırarak ekrana "Girilen deger dizinin 6. elemanidir" cümlesini yazdıracaktır. Diğer yandan kullanıcı 30 değerini girerse; dizinin herhangi bir elemanı 30 değerine eşit olmadığından *Arama(diz, boyut, x)* fonksiyonu verilen girdi değerleri için 0 değerine dönecek ve 'Arama(diz, boyut, x) == 0' koşulu doğru olacaktır. Dolayısıyla program if.. else kontrol yapısının gövdesindeki kodu çalıştırarak ekrana "Dizi girilen tamsayi degerini icermemektedir " cümlesini yazdıracaktır.

```
    int diz[15] = {7, 12, 6, 21, 17, 5, 12, 32, 9, 2, 16, 31,
3, 18, 23};
    int x;
    cout << "Pozitif bir tamsayi degeri giriniz : " <<
endl;
    cin >> x;
    int boyut = (sizeof(diz)/sizeof(*diz));
    if(Arama(diz, boyut, x) == 0)
        cout << "Dizi girilen tamsayi degerini icer-
memektedir" << endl;
    else
        cout << "Girilen deger dizinin " << Arama(-
diz, boyut, x) << ". elemanidir" << endl;
    }
```

### Dizilerde Sıralama

Burada tanımlı dizilerin elemanlarını küçükten büyüğe sıralamamıza imkan sağlayan sıralama algoritmalarına yer verilecektir. Algoritmalara geçmeden, önce diziler için sıralama problemini tanımlayalım: k dizinin boyutu olmak üzere kıyaslanabilir ve sıralanabilir elemanlardan oluşan tanımlı bir dizi[k] dizisi için sıralama problemi, dizinin elemanlarını dizi[0] ≤ dizi[1] ≤ ... ≤ dizi[k-1] şartını sağlayacak şekilde yeniden sıralamak şeklinde tanımlanmaktadır. Dizilerin sıralı bir şekilde tutulmasının daha kullanışlı bir veri yapısı oluşturması, dizilerde sıralamanın arama algoritmaları gibi farklı algoritmaların alt yordamı olması ve sıralama problemine etkin çözüm geliştirmek için yapılan uğraşın algoritmik tasarım açısından zengin ve kullanışlı bir literatür oluşturuyor olması gibi sebeplerden ötürü sıralama problemi bilgisayar bilimlerindeki en temel problemlerden biri olagelmiştir.

Dizilerin elemanlarını küçükten büyüğe sıralamak mevzu olduğunda akla gelen en aşikar ve kolay yaklaşım şüphesiz küçük elemanların bulunarak dizinin ilk sıralarına yerleştirilmesi ya da büyük elemanların bulunarak dizinin son sıralarına yerleştirilmesi olacaktır. Bu yaklaşımlardan ilki seçmeli sıralama (selection sort) ismiyle literatüre girmiş bir algoritmayken, ikincisi kabarcık sıralaması (bubble sort) ismiyle literatürde yer bulmuştur. Aşağıdaki kod bloğu küme parantezi yöntemi kullanılarak tanımlanmış 10 uzunluklu bir tam sayı dizisinin elemanlarını kabarcık sıralaması

yöntemi kullanarak sıralamaktadır. Kod bloğundaki dıştaki for döngüsünün ilk adımında (i = 0 için) dizinin en büyük elemanı içteki for döngüsü yardımıyla dizinin son indisine taşınmaktadır. Dıştaki for döngüsünün ikinci adımında ise dizinin en büyük ikinci elemanı içteki for döngüsü yardımıyla dizinin sondan bir önceki indisine taşınmaktadır. Bu şekilde dizinin büyük elemanları dizinin sonlarına doğru taşınarak nihayetinde elemanları küçükten büyüğe sıralı bir dizi elde edilmektedir.

```
int diz[10] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
int boyut = (sizeof(diz)/sizeof(*diz));
int temp;
 for(int i = 0; i < boyut - 1; i++){
    for(int j = 0; j < boyut - i; j++){
       if(diz[j] > diz[j+1]){
          temp = diz[j+1];
          diz[j+1] = diz[j];
          diz[j] = temp;
       }
    }
}
```

**UYGULAYALIM**

Yukarıda verilen tamsayı dizisini Seçmeli Sıralama Algoritması yardımıyla sıralayan bir program geliştirin.

## İKİ BOYUTLU DİZİLER

Matematikteki matris kavramının C++ programlama dilindeki karşılığı olarak değerlendirilebilecek iki boyutlu diziler aşağıdaki kod satırı üzerinden tanımlanmaktadır.

```
veri_turu diziAdi[boyut_1][boyut_2];
```

Tek boyutlu dizilerde olduğu gibi *veri_turu* bellekte saklanacak verilerin türünü, *boyut_1* tanımlanan dizinin satır sayısını ve *boyut_2* ise dizinin sütun sayısını belirtmektedir. Kod satırı çalıştırıldığı anda program *boyut_1*boyut_2*veri_turu* boyutu kadarlık alanı dizi için ayırmaktadır. Örneğin Şekil 9.2'deki matris iki boyutlu bir diziyi göstermektedir. Şekilden de anlaşılacağı üzere iki boyutlu diz dizisinin her elemanı, i ve j elemanı belirleyen indisler olmak üzere diz[i][j] şeklinde gösterilmektedir.

| | Sütun 0 | Sütun 1 | Sütun 2 |
|---|---|---|---|
| Satır 0 | diz[0][0] | diz[0][1] | diz[0][2] |
| Satır 1 | diz[1][0] | diz[1][1] | diz[1][2] |
| Satır 2 | diz[2][0] | diz[2][1] | diz[2][2] |

*Şekil 9.3 İki Boyutlu Dizinin Matris ile Gösterimi.*

İki boyutlu diziler bellekte tek boyutlu bir dizi olarak tutulmaktadırlar. Bu açıdan bakıldığında iki boyutlu diziler, her elemanı ayrı bir dizi olan tek boyutlu diziler olarak değerlendirilebilirler. Dolayısıyla iki boyutlu dizilerin satır ve sütün indisleri üzerinden yorumlanması bir kabule dayanmaktadır.

C++ programlama dilinde tanımlanan bir dizinin elemanlarına, elemanların matris içerisindeki konumu belirten satır ve sütun indislerinin sırasıyla alt simge işleçleri [ ] içerisine yazılmasıyla erişim sağlanmaktadır. Örneğin aşağıdaki kod satırı üzerinden iki satırlı ve iki sütunlu iki boyutlu bir tamsayı dizisi tanımlanmaktadır. Tanımlı bu dizinin ikinci satır birinci sütunundaki elemanına *dizi[1][0]* kodu yardımıyla erişim sağlanmaktadır. Dolayısıyla aynı sayı dizisinin birinci satırındaki elemanlar sırasıyla *dizi[0][0]* ve *dizi[0][1]* olacakken, ikinci satırındaki elemanlar *dizi[1][0]* ve *dizi[1][1]* olacaktır.

```
int dizi[2][2];
```

Tek boyutlu dizilerde olduğu gibi iki boyutlu dizilerde de tanımlama esnasında dizinin elemanlarına değer atama işlemi küme parantezi yoluyla yapılabilmektedir. Küme parantezi ile değer atama işleminde atanacak değerler her satır için ayrı parantez kullanılarak tanımlanmaktadır. Örneğin aşağıdaki kod satırında üç satırlı ve üç sütunlu bir tamsayı dizisi tanımlanmış ve küme parantezi yardımıyla dizinin elemanlarına 1'den 9'a kadar tamsayı değerleri sırasıyla atanmıştır.

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

İki boyutlu dizilerde küme parantezi yoluyla değer atama işlemi, aşağıda gösterildiği gibi her satır için ayrı bir parantez kullanılmaksızın da gerçekleştirilebilmektedir. Küme parantezi yardımıyla atama işleminde dizinin boyutundan daha az sayıda değer girilmişse, program kalan elemanlara tam sayılarda tanımlanmış diziler için 0 değerini atayacaktır. Dolayısıyla aşağıdaki kod satırı üzerinden tanımlanmış tamsayı dizisinin elemanları sırasıyla *dizi[0][0] = 1, dizi[0][1] = 2, dizi[0][2] = 3, dizi[1][0] = 4, dizi[1][1] = 5, dizi[1][2] = 6, dizi[2][0] = 7, dizi[2][1] = 0 ve dizi[2][2] = 0* olacaktır.

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

Diğer yandan tanımlı iki boyutlu dizilerin elemanlarına iç içe döngüler yardımıyla erişim sağlanabilmektedir. Aşağıdaki kod bloğunda ise tanımlı bir dizinin elemanlarına iç içe iki for döngüsü yardımıyla nasıl değer atandığı gösterilmektedir. İç içe döngülerin ilkindeki 'diz[i][j] = i + j' kod satırı üzerinden dizinin elemanlarına satır ve sütun indislerinin toplamı değer olarak atanmaktadır. İç içe döngülerin ikincisindeki 'cout << "diz[" << i << "][" << j << "]="<< diz[i][j] << " " ' kod satırı yardımıyla dizinin her bir elemanının güncel değeri ekrana yazdırılmaktadır. Dolayısıyla kod bloğu çalıştırıldığında oluşacak ekran görüntüsü Şekil 9.4'deki gibi olacaktır.

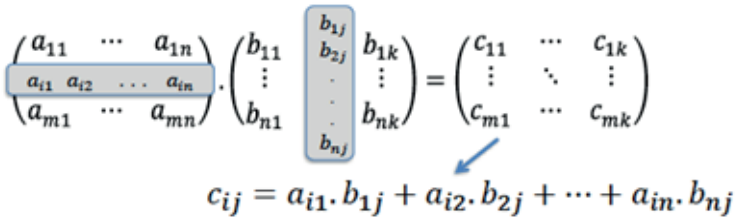
```
int diz[3][3];
 for (int i = 0 ; i < 3; i++){
    for (int j = 0; j < 3; j++){
       diz[i][j] = i + j;
    }
 }
 for (int i = 0 ; i < 3; i++){
    for (int j = 0; j < 3; j++){
       cout << "diz[" << i << "][" << j << "]="<< diz[i]
[j] << " " ;
    }
    cout << "" << endl;
 }
```

```
diz[0][0]=0 diz[0][1]=1 diz[0][2]=2
diz[1][0]=1 diz[1][1]=2 diz[1][2]=3
diz[2][0]=2 diz[2][1]=3 diz[2][2]=4
```

*Şekil 9.4 İki Boyutlu Tamsayı Dizisinin Elemanlarının Ekrana Yazdırılması.*

## İki Boyutlu Dizilerin Çarpımı

Hatırlanacağı gibi verilen iki matrise çarpma işleminin uygulanabilmesi için, ilk matrisin sütun sayısı ile ikinci matrisin satır sayısının birbirine eşit olması gerekmektedir. Bu kısıt sağlandığında verilen m satır ve n sütun içeren A matrisi ile n satır ve k sütun içeren B matrisinin çarpımı aşağıda gösterildiği şekilde olmaktadır:



$$c_{ij} = a_{i1}.b_{1j} + a_{i2}.b_{2j} + \cdots + a_{in}.b_{nj}$$

*Şekil 9.5 Matrislerin Çarpımı.*

Şekil 9.5'ten de anlaşılacağı gibi m satır ve k sütunlu C sonuç matrisinin her bir cij elemanı A matrisinden i. satır ve B matrisinden j. sütunun elemanları yukarıda gösterildiği gibi bir işleme tabi tutularak bulunmaktadır. Aşağıdaki kod bloğu içe içe üç for döngüsü yardımıyla verilen iki matrisin çarpımını hesaplamaktadır. Kod bloğunda önce 2 satır ve 3 sütunlu iki boyutlu bir tamsayı dizisi *AMatrisi* tanımlanmış ve bu matrisin satırlarına sırasıyla {1, 2, 3} ve {4, 5, 6} değerleri atanmıştır. Yine benzer şekilde 3 satır ve 2 sütunlu iki boyutlu bir tamsayı dizisi *BMatrisi* tanımlanmış ve bu matrisin satırlarına sırasıyla {1, 2,}, {3, 4} ve {5, 6} değerleri atanmıştır. Son olarak çarpım matrisini saklayacak 2 satır ve 2 sütunlu iki boyutlu bir tamsayı dizisi *CMatrisi* tanımlanmış ve bu matrisin elemanlarına 0 değeri atanmıştır. Kod bloğundaki dıştaki iki for döngüsü sırasıyla sonuç matrisinin satır ve sütun indislerini tararken, içteki for döngüsü yukarıda verilen

'cij = ai1.b1j + ai2.b2j + ... + ain.bnj'

formülünü uygulayarak sonuç matrisinin elemanlarının alacağı değerleri hesaplamaktadır.

```
int AMatrisi[2][3] = {{1,2,3},{4,5,6}};
int BMatrisi[3][2] = {{1,2},{3,4},{5,6}};
int CMatrisi[2][2] = { };
for (int i = 0; i < 2; i++){
   for (int j = 0; j < 2; j++){
      for (int n = 0; n < 3; n++){
            CMatrisi[i][j] += AMatrisi[i][n] * BMatrisi[n][j];
      }
   }
}
for (int i = 0 ; i < 2 ; i++){
    for (int j = 0; j < 2; j++){
       cout << "CMatrisi[" << i << "][" << j << "]="<< CMatrisi[i][j] << " ";
    }
    cout<<""<<endl;
}
```

Yukarıdaki kod bloğunda iç içe döngülerin ikincisi ise 'cout << "CMatrisi[" << i << "][" << j << "]="<< CMatrisi[i][j] << " "; ' kod satırı üzerinden sonuç matrisinin elemanlarını ekrana yazdırmaktadır. Kod bloğu çalıştırıldığında oluşacak ekran görüntüsü Şekil 9.6'daki gibi olacaktır.

```
CMatrisi[0][0]=22 CMatrisi[0][1]=28
CMatrisi[1][0]=49 CMatrisi[1][1]=64
```

*Şekil 9.6 İki Boyutlu Dizilerin Çarpımı.*

### İki Boyutlu Dizilerin Fonksiyonlara Aktarılması

Burada fonksiyonlar yardımıyla satır ve sütun sayıları eşit kare bir matrisin transpozuna eşit olup olmadığı, diğer bir ifadeyle matrisin köşegenine göre simetrik olup olmadığını belirleyen bir program incelenecektir. Aşağıda tanımı verilen Simetrik fonksiyonu satır ve sütun sayısı eşit iki boyutlu bir diziyi ve dizinin satır sayısını girdi olarak almakta; ve iç içe iki for döngüsü yardımıyla bütün satır ve sütun indisleri için 'dizi[i][j] = dizi[j][i]' eşitliğinin sağlanıp sağlanmadığını kontrol ederek verilen iki boyutlu dizinin simetrik olup olmadığına karar vermektedir. Fonksiyon dizi simetrikse 1 değerine, simetrik değilse 0 değerine dönmektedir. Tek boyutlu dizilerden farklı olarak, iki boyutlu diziler için tanımlanan fonksiyonlarda ikinci boyut mutlaka belirtilmelidir. Dikkat edilirse; aşağıdaki fonksiyon tanımında dizinin ikinci boyutu 4 olarak tanımlanmıştır.

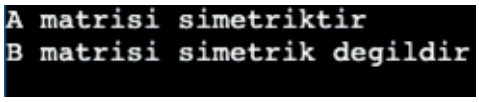
```
int Simetrik(int dizi[][4], int boyut) {
   for (int i = 0; i < boyut; i++){
      for (int j = 0; j < boyut; j++){
         if(dizi[i][j] != dizi[j][i]){
            return 0;
         }
      }
   }
   return 1;
}
```

**DİKKAT EDELİM**

Çok boyutlu diziler üzerinde fonksiyon tanımlanırken, ilk boyut dışındaki diğer boyutlar mutlaka belirtilmelidir.

Aşağıda verilen kod bloğu ise yukarıda tanımlanan Simetrik fonksiyonu yardımıyla iki boyutlu tanımlı A ve B tamsayı dizilerinin simetrik olup olmadığını belirlemektedir. Örneğin tanımlı dizilerden ilki simetrik olduğundan 'Simetrik(A, 4) == 1' koşulu doğru olacak ve program ilk if...else kontrol yapısının gövdesindeki kodu çalıştırarak ekrana "A matrisi simetriktir" cümlesini yazdıracaktır. Diğer yandan ikinci dizi için 'A[0][1] == A[1][0]' ifadesi doğru olmadığından 'Simetrik(B, 4) == 1' koşulu doğru olmayacak ve program ikinci if...else kontrol yapısının alternatif gövdesindeki kodu çalıştırarak ekrana "B matrisi simetrik degildir" ifadesini yazdıracaktır. Dolayısıyla aşağıdaki kod bloğu çalıştırıldığında ekran çıktısı Şekil 9.7'deki gibi olacaktır.

```
int A[4][4] = {{1,2,1,2},
               {2,1,2,1},
               {1,2,1,2},
               {2,1,2,1}};
int B[4][4] = {{1,2,1,2},
               {3,1,2,1},
               {1,2,1,2},
               {2,1,2,1}};
if(Simetrik(A, 4) == 1){
    cout << "A matrisi simetriktir" << endl;
}
else {
    cout << "A matrisi simetrik degildir" << endl;
}
if(Simetrik(B, 4) == 1){
    cout << "B matrisi simetriktir" << endl;
}
else {
    cout << "B matrisi simetrik degildir" << endl;
}
```




*Şekil 9.7 Matrislerin Simetrik Olup Olmadığını Belirleyen Kodun Ekran Çıktısı.*

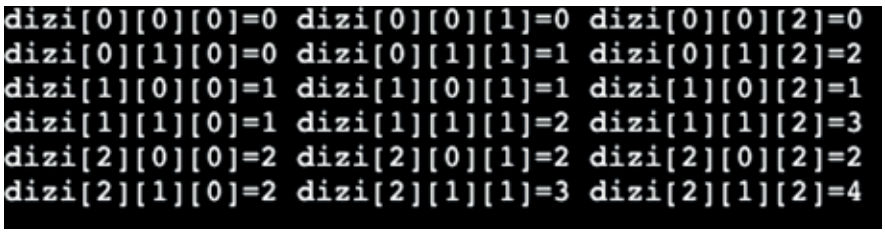
## ÇOK BOYUTLU DİZİLER

C++ programlama dili ikiden fazla boyuta sahip diziler de tanımlanabilmesi imkan vermektedir. İki boyutlu dizilere benzer şekilde çok boyutlu diziler de aşağıda gösterildiği gibi dizinin farklı boyutlardaki eleman sayıları alt simge işleci içerisinde belirtilerek tanımlanmaktadır.

veri_turu diziAdi[boyut_1][boyut_2]...[boyut_n];

Örneğin aşağıdaki kod bloğunda, birinci boyutunda 3, ikinci boyutunda 2 ve üçüncü boyutunda 3 eleman barındıran üç boyutlu bir tamsayı dizisi tanımlanmış; ve bu dizinin elemanlarına i, j ve k sırasıyla birinci, ikinci ve üçüncü boyuttaki indisler olmak üzere 'i + j * k' formülü kullanılarak iç içe üç for döngüsü yardımıyla tamsayı değerleri atanmıştır. Dolayısıyla kod bloğu çalıştırıldığında ekran çıktısı Şekil 9.8'deki gibi olacaktır.

```
int dizi[3][2][3] = { };
for (int i = 0; i < 3; i++){
   for (int j = 0; j < 2; j++){
      for (int k = 0; k < 3; k++){
         dizi[i][j][k] = i + j * k;
         cout << "dizi[" << i << "][" << j << "][" << k << "]=" << dizi[i][j][k] << " ";
      }
      cout << "" << endl;
   }
}
```




*Şekil 9.8 Üç Boyutlu Tamsayı Dizisinin Elemanlarının Ekrana Yazdırılması.*

## BÖLÜM ÖZETİ

Bu bölümde aynı veri türünden belirli sayıda elemanı bellekte sıralı bir şekilde saklamaya imkan veren dizilerden bahsedilmiştir. Dizilerde saklanacak elemanların sonlu sayıda ve aynı veri türünde olması gerekmektedir. Dizilerdeki elemanlara erişim ilgili elemanların bellekteki sırasını belirten indisler yoluyla yapılmaktadır.

Dizinin elemanlarına değer atama işlemi dizi tanımlandığında küme parantezi yardımıyla yapılabileceği gibi, dizi tanımlandıktan sonra ihtiyaca göre kullanıcıdan gelen girdiler veya program içerisinde uygulanan işlemler üzerinden atama işleci yardımıyla da gerçekleştirilebilmektedir. Diğer yandan tanımlamış karakter dizisinin elemanlarına değer atama işlemi, tam sayılarda olduğu küme parantezi yardımıyla yapılabileceği gibi atanacak değerlerin çift tırnak içinde tek bir kelime olarak yazılması ile de mümkündür.

Bunların yanı sıra dizi kavramını pekiştirmek ve dizilerin farklı problemlerin çözümünde nasıl kullanılabildiğini göstermek amacıyla dizilerde arama ve sıralama algoritmaları da incelenmiştir. Dahası dizilerin fonksiyonlara aktarılmasının nasıl yapıldığı örnekler üzerinden anlatılmıştır.

Bölümde ayrıca matematikteki matris kavramının C++ programlama dilindeki karşılığı olarak değerlendirilebilecek iki boyutlu dizilerden de bahsedilmiştir. Ayrıca iki boyutlu dizi kavramını pekiştirmek adına, iki boyutlu diziler kullanılarak verilen iki matrisin çarpımını hesaplayan bir uygulamaya da yer verilmiştir.

## GÖZDEN GEÇİRELİM

**1. Aşağıdakilerden hangisinde dizi doğru şekilde tanımlanmıştır?**

a) int dizi[5] = {5, 3, 1, 10, 8, 7};
b) double dizi[5] = {3.2, 0.5, c};
c) int dizi = {5, 2, 1, 3, 4,};
d) char dizi[5];
e) char dizi[5] = {a, d, a, n, a};

2. Aşağıdakilerden hangisinde dizi yanlış tanımlanmıştır?

a) int dizi[5] = {10, 6, 5, 7};
b) bool dizi[5];
c) char dizi[5] = 'c';
d) char dizi[5] = "c";
e) float dizi[5];

**3. Aşağıdakilerden hangisinde iki boyutlu tamsayı dizisi doğru şekilde tanımlanmıştır?**

a) int dizi[2][4] = {{4, 4, 4, 4}, 3, 3, 3, 3};
b) int dizi[2];
c) int dizi = {{4, 4, 4, 4}, {3, 3, 3, 3}};
d) int dizi[2][4] = {{4, 4, 4, 4, 3, 3, 3, 3, 3}};
e) int dizi[2][4] = {4, 4, 4, 4, 3, 3, 3, 3};

**4. Aşağıdakilerden hangisinde iki boyutlu dizi yanlış tanımlanmıştır?**

a) int dizi[2][4] = {{2, 2, 2, 2}, {4, 4, 4, 4}};
b) bool dizi[2][4];
c) char dizi[2][4] = {"ada", "na"};
d) char dizi[2][2] = "adana";
e) int dizi[2][2] = {1, 3, 5};

**5.**

```
int dizi[6] = {1, 2};
for(int i = 2; i < 6; i++){
      dizi[i] = 2*dizi[i - 1] - dizi[i - 2];
}
```

**Yukarıdaki kod bloğu çalıştırıldıktan sonra dizinin son elemanının değeri aşağıdakilerden hangisi olacaktır?**

a) 3 b) 4
c) 5 d) 6 e) 7

**6.**

```
int dizi[5] = {1, 2};
for(int i = 2; i < 5; i++){
      if(i%2 ==1)
            dizi[i] = 2*dizi[i - 1] - dizi[i - 2];
      else
            dizi[i] = 3*dizi[i - 2] - dizi[i - 1];
}
```

**Yukarıdaki kod bloğu çalıştırıldıktan sonra dizinin son elemanının değeri aşağıdakilerden hangisi olacaktır?**

a) 0 b) 1
c) 2 d) 3
e) 4

**7.**

```
char dizi[8] = "adana";
dizi[1] ='t';
dizi[3] ='\0';
cout<<dizi<<endl;
```

**Yukarıdaki kod bloğu çalıştırıldığında ekran çıktısı aşağıdakilerden hangisi olacaktır?**

a) adana
b) ada\0
c) ada
d) ata\0
e) ata

**8.**

```
int diz[12] = {25, 13, 10, 21, 8, 7, 17, 1, 4, 5, 19, 18};
int indis = 0, x = 5;
int boyut = (sizeof(diz)/sizeof(*diz));
while (indis < boyut){
      if (diz[indis] == x){
         cout << "Aranan tam sayi degeri dizinin "
<< indis + 1 << ". elemanidir" << endl;
         break;
      }
      indis++;
}
if(indis >= boyut)
      cout << "Dizi aranan tam sayi degerini icermemektedir" << endl;
```

**Yukarıdaki kod bloğu çalıştırıldığında ekrana yazdırılacak olan ifade aşağıdakilerden hangisi olacaktır?**

a) Aranan tam sayi degeri dizinin 9. elemanıdır
b) Aranan tam sayi degeri dizinin 10. elemanıdır
c) Aranan tam sayi degeri dizinin 11. elemanıdır
d) Aranan tam sayi degeri dizinin 12. elemanıdır
e) Dizi aranan tam sayi degerini icermemektedir

**9.**

```
int diz[12] = {25, 13, 10, 21, 8, 7, 17, 1, 4, 5, 19, 18};
int temp;
for(int i = 0; i < (sizeof(diz)/sizeof(*diz))-1; i++){
      for(int j = 0; j < (sizeof(diz)/sizeof(*diz))-1; j++){
         if(diz[j] < diz[j+1]){
            temp = diz[j];
            diz[j] = diz[j+1];
            diz[j+1] = temp;
         }
      }
}
for(int i = 0; i < (sizeof(diz)/sizeof(*diz)); i++){
      cout << diz[i] << " ";
}
```

**Yukarıdaki kod bloğu çalıştırıldığında ekrana yazdırılacak olan ifade aşağıdakilerden hangisi olacaktır?**

a) 25 13 10 21 8 7 17 1 4 5 19 18
b) 1 4 5 7 8 10 13 17 18 19 21 25
c) 25
d) 25 21 19 18 17 13 10 8 7 5 4 1
e) 1

**10.**

```
int diz[4][4];
for (int i = 0 ; i < 4; i++){
    for (int j = 0; j < 4; j++){
        diz[i][j] = i * j;
    }
}
```

**Yukarı kod bloğu çalıştırıldıktan sonra dizinin 3. satır ve 4. sütunundaki elemanın alacağı değer aşağıdakilerden hangisi olacaktır?**

a) 1
b) 2
c) 6
d) 12
e) 16

**Yanıt Anahtarı**

| | | | | |
|---|---|---|---|---|
| 1.d | 2.c | 3.e | 4.d | 5.d |
| 6.d | 7.e | 8.b | 9.d | 10.c |