Bölüm 07

Array

Array Nedir?
Array Yaratma
[ ] Operatörü
Array'in Bileşenleri
Seçkili (random) Erişim
Array Türleri
    Bir Boyutlu Array
    Çok Boyutlu Array
    Array Arrayi (çentikli array)

Array Nedir?

Array, aynı tipten çok sayıda değişken tanımlamak için kullanılır. Soyut bir veri yapısıdır. Matematikteki sonlu dizi kavramına benzer. C# dilinde array bir sınıftır. Her sınıfın soyut bir veri tipi olduğunu biliyoruz. Array sınıfı array yaratma, arraylerle işlem yapma, array içinde bileşen arama ve array'in bileşenlerini sıralama gibi array ile ilgili işlemleri yapmaya yarayan öğeleri içeren bir sınıftır.

Array Yaratma

Array bir sınıf olduğuna göre, bir array'i yaratma bir sınıftan nesne yaratma gibidir. Üç aşaması vardır:

```
Birinci Aşama    : Array sınıfının bildirimi
İkinci Aşama     : Array sınıfından array nesnesini yaratma
Üçüncü Aşama     : Array'in bileşenlerine değer atama
```

Şimdi bu üç aşamanın nasıl yapıldığını bir örnek üzerinde görelim.

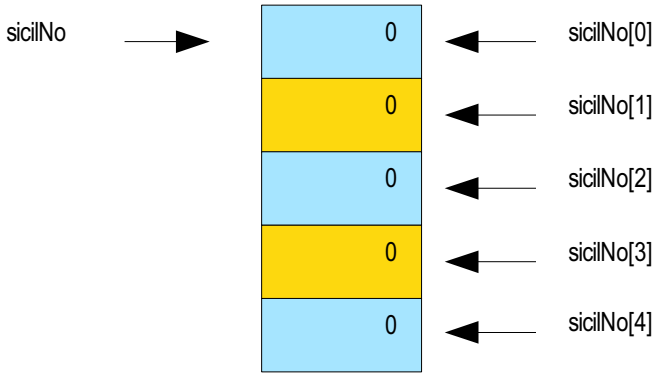
Birinci aşama:

```
int []    sicilNo ;
```

deyimi int tipinden `sicilNo` adlı bir array bildirimidir. Bu aşamada `sicilNo` null işaret eden bir referanstır (işaretçi, pointer).

İkinci aşama:

```
sicilNo = new int[5] ;
```



kurucu (constructor) deyimi `sicilNo` tarafından işaret edilen bir nesne yaratır. Başka bir deyişle, ana bellekte 5 tane `int` değer tutacak bir yer ayırır. `SicilNo` bellekte ayrılan bu yeri işaret eder. O nedenle, `sicilNo`'ya referans (işaretçi, pointer) denmektedir. O halde, arrayler referans tipidir. Başka bir deyişle, array'in kendisi bir değer değil, kendisine ait nesnenin bellekte bulunduğu adresi işaret eden referanstır.

Arrayin işaret ettiği nesnenin içinde `int` tipi veri tutacak 5 tane bellek adresi vardır. Bu adresler

SicilNo[0]

sicilNo[1]

sicilNo[2]

sicilNo[3]

sicilNo[4]

adları tarafından işaret (referans) edilirler.

[] Operatörü

Array adının sonuna gelen `[]` parantezleri, arrayin bileşenlerini, yukarıda gösterildiği gibi, indisleriyle (damga, numara) belirler.

Array'in Bileşenleri

`sicilNo[i]` `(i=0,1,2,3,4)` ler söz konusu nesne içinde birer değişkendir. Bu değişkenler sayesinde, array beş tane int tipi veriyi tutabilme yeteneğine sahip olur. Bu değişkenlere array'in bileşenleri denir. `0,1,2,3,4` sayıları bileşenlerin sıra numaralarıdır; damga (index) adını alırlar. Sıra numaraları (index) daima 0 dan başlar, birer artarak gider. n tane bileşeni olan bir array'in ilk bileşeninin damgası 0, son bileşeninin damgası (n-1) olur. Bir array'in uzunluğu onun bileşenlerinin sayısıdır.

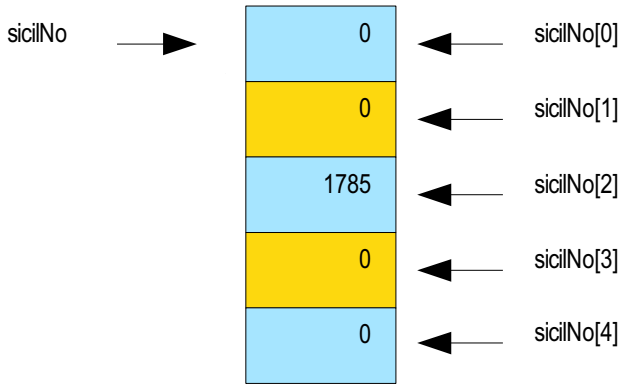
Eğer `new int[5]` yerine new `int[500]` yazsaydık, 5 bileşen yerine 500 bileşen elde ederdik.

Arrayin işaret ettiği nesne yaratılınca, onun bileşenleri kendiliğinden başlangıç değeri alırlar. Bunlara `öndeğer` (default value) denir. Öndeğerler bileşenlerin veri tipine bağlı olarak değişir. C# dilinde bütün sayısal değişkenlerin öndeğerleri daima 0 olur. Referans tiplerde ise `null` olur. O nedenle, yukarıda `sicilNo` referansının işaret ettiği nesne içindeki `SicilNo[0]`, `sicilNo[1]`, `sicilNo[2]`, `sicilNo[3]` , `sicilNo[4]` bileşenlerinin (değişken) öndeğerleri kendiliğinden 0 olur.

Üçüncü aşama:

```
sicilNo[2] = 1785;
```

ataması, array'in üçüncü bileşenine 1785 değerini atar.



İstenirse öteki bileşenlere de benzer yolla atamalar yapılabilir.

Yukarıdaki üç aşamayı birleştirerek iki ya da bir adımda hepsini bitirebiliriz. Örneğin,

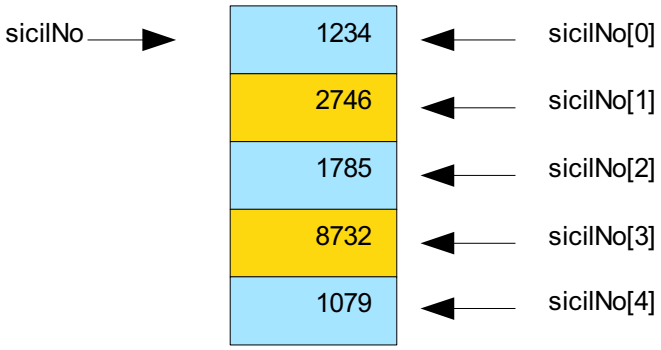
```
int [] sicilNo = new int[] {1234, 2746, 1785, 8732, 1079};
```

deyimi üç aşamayı birden yapar.

İstenirse,

```
int [] sicilNo = new int[5] ;
```

deyimi ile ilk iki aşama tamamlanır, bileşenlere değer atama işi sonraya bırakılabilir.



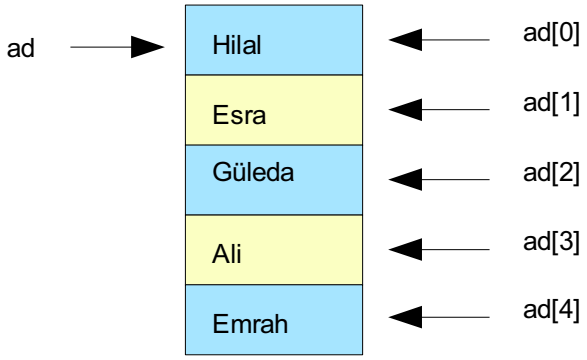
C# dilinde her veri tipinden array yaratılabilir. Örneğin,

```
string [] ad = new string[] {"Hilal", "Esra", "Güleda", "Ali", "Emrah"};
```

deyimi `string` tipinden bir array sınıfı bildirmiş, onun bir nesnesini yaratmış ve o nesnenin bileşenlerine `string` tipi değerler atamıştır. Bu atama

```
ad[0] = "Hilal" ;  ad[1] = "Esra" ;  ad[2] = "Güleda" ;  ad[3] = "Ali" ;
ad[4] = "Emrah" ;
```

atamalarına denktir.



Artık, array yaratmak için genel sözdizimini yazabiliriz:

```
veriTipi []  arrayAdı ;    (array bildirimi)
arrayAdı = new veriTipi [bileşen sayısı];     (array nesnesini yaratma)
```

Array nesnesi yaratıldıktan sonra, bileşenlerine değer atama eylemi, değişkenlere değer atama gibidir. İstenen bileşen indeks sayısı ile seçilerek seçkili (random) değer atanabilir.

Array bildiriminde, yukarıda yaptığımız gibi, arrayin uzunluğunu (bileşenlerinin sayısını) nesneyi yaratırken belirleyebiliriz. Buna sabit uzunluk belirleme diyeceğiz. Ancak, bazı durumlarda, nesneyi yaratırken arrayin uzunluğunu tam bilmiyor olabiliriz. Bu durumda, arrayin uzunluğunu dinamik olarak belirleriz. Dinamik uzunluklu arraye bileşen ekledikçe arrayin uzunluğu kendiliğinden artar. İstendiği an dinamik uzunluklu array sabit uzunluklu array haline getirilebilir. Örneğin,

```
short []  sıraNo ;
```

bildirimi dinamik uzunluklu bir array tanımlar.

```
double []  aylıkÜcret[145];
```

deyimi 145 bileşenli bir array tanımlar.

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

deyimi 10 bileşenli bir array yaratır.

```
int[] arr;
arr = new int[10] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
```

deyimi 10 bileşenli bir array yaratır ve bileşenlerine değerler atar . Atanan değer sayısı bileşen sayısına eşit olmalıdır. Ayrıca, bileşenlere değerlerin yazılış sırasıyla atandığını unutmayınız. Arrayin belirli bir bileşenine değer atamak isterseniz, Seçkili Erişim paragrafına bakınız.

Array bileşenlerine değer atama eylemi, genellikle program koşarden dinamik olarak yapılır. Aşağıdaki örnek, bir döngü ile array bileşenlerinin bazılarına değer atamaktadır. Sonra bütün bileşenlerin değerlerini konsola yazdırmaktadır. Değer atanmamış bileşenlerin (default) değerlerinin 0 olduğuna dikkat ediniz.

Array.cs

```
using System;

namespace Arrayler
{
    class Array05
    {
        static void Main()
        {
            int[] çarpan = new int[10];
            for (int i = 6; i < çarpan.Length; i++)
            {
                çarpan[i] = i * i;
            }
            for (int i = 0; i < çarpan.Length; i++)
            {
                Console.Write(çarpan[i]);
                Console.Write("\t");
            }
        }
    }
}
```

Aşağıdaki program da arrayin bileşenlerine bir döngü ile değer atamakta ve atanan değerleri başka bir döngü ile konsola yazdırmaktadır.

Array.cs

```
using System;

class Array01
{
    static void Main()
    {
        int[] intSayı = new int[5];
        for (int i = 0; i < intSayı.Length; i++)
            intSayı[i] = i * 10;
        for (int i = 0; i < intSayı.Length; i++)
            Console.WriteLine("intSayı[{0}] = {1}", i, intSayı[i]);
    }
}
```

Çıktı

```
intSayı[0] = 0
intSayı[1] = 10
intSayı[2] = 20
intSayı[3] = 30
intSayı[4] = 40
```

Seçkili (random) Erişim

Arrayin bileşenleri değişken olduklarından, onlara istendiğinde değer atanabileceğini, istenirse atanan değerlerin değiştirilebileceği açıktır. Yukarıdaki sicilNo[2] = 1785 atama deyimi, arraylerin üstün bir niteliğini ortaya koyar. Arrayin istenen bileşenine indeks sayısı ile doğrudan erişmek mümkündür. Her bileşen bir değişken olduğu için, o bileşen tipiyle yapılabilen her işlem onlar için de yapılabilir, değişkenlerle ilgili kurallar bileşenler için de aynen geçerlidir. SicilNo[2] = 1785 deyimi indeksi 2 olan bileşene 1785 değerini atamıştır. x aynı tipten bir değişken ise

```
x = sicilNo[2];
```

ataması, sicilNo[2] bileşeninin değerini x değişkenine aktarır; öyleyse, bu atama deyimi

```
x = 1785 ;
```

atamasına denktir. Aşağıdaki örnek, bileşenlerle işlem yapılabileceğini göstermektedir. Aylık ücretler ve gelir vergileri iki ayrı array ile tutulmakta, bir döngü ile %30 gelir vergisi hesaplanıp konsola yazılmaktadır.

```
using System;

namespace Arrayler
{
    class Array01
    {
        static void Main()
        {
            double[] aylıkÜcret = new double[3];
            aylıkÜcret[0] = 2782.45;
            aylıkÜcret[1] = 9346.74;
            aylıkÜcret[2] = 10867.83;

            double[] gelirVergisi = new double[3];
            for (int i = 0; i < aylıkÜcret.Length; i++)
            {
                gelirVergisi[i] = aylıkÜcret[i] * 30 / 100;

            Console.WriteLine("{0:c}  ücretin gelir vergisi =  {1:c} "
, aylıkÜcret[i], gelirVergisi[i]);
            }
        }
    }
}
```

Çıktı

```
2.782,45   TL  ücretin gelir vergisi =  834,74 TL
9.346,74   TL  ücretin gelir vergisi =  2.804,02 TL
10.867,83 TL  ücretin gelir vergisi =  3.260,35 TL
```

Bileşenlerden başka değişenlere veri aktarılabilir. Bileşenlerden veri aktarmalarında *istemli (explicit)* ve *istemsiz (implicit)* dönüşüm kuralları aynen geçerlidir. Aşağıdaki program `short` tipinden `int` tipine *istemsiz dönüşüm* yapılabileceğini göstermektedir.

Array.cs

```
using System;

namespace Arrayler
{
    class Array01
    {
        static void Main()
        {
            short[]  a = { 1, 2, 3, 4 };
            int[]  b = { 5, 6, 7, 8, 9 };
            Console.WriteLine(b[2]);
           b[2] = a[3];
            Console.WriteLine("b[2] = {0}" ,b[2]);
        }
    }
}
```

Çıktı

```
b[2] = 7
b[2] = 4
```

Aynı aktarmayı tersinden yapmak isteyelim. Yukarıdaki programda b[2] = a[3] deyimi yerine a[2] = b[3] yazarsak, derleyiciden şu hata mesajını alırız:

```
Error   1   Cannot implicitly convert type 'int' to 'short'. An explicit conversion exists (are you missing a cast?)...
```

Derleyicinin *istemsiz (implicit)* yapmadığı dönüşümü zorla yaptırmak istersek, *istemli dönüşüm* (implicit casting) yapabiliriz:

```
using System;

namespace Arrayler
{
    class Array01
    {
        static void Main()
        {
            short[] a = { 1, 2, 3, 4 };
            int[] b = { 5, 6, 7, 8, 9 };
            Console.WriteLine("a[2] = {0}", a[2]);
            a[2] = (short)b[3];
            Console.WriteLine("a[2] = {0}" , a[2]);
        }
    }
}
```

Çıktı

```
a[2] = 3
a[2] = 8
```

Arrayler çok sayıda değişken tanımlama ve bileşenlerine doğrudan istemli erişim sağlamaları yanında, döngüleri de çok kolaylaştırırlar. Aşağıdaki program `foreach` döngüsünün arraylere nasıl uygulanacağını göstermektedir.

Array.cs

```
using System;

namespace Arrayler
{
    class Array01
    {
        static void Main()
        {
            int[] arr = { 11, 12, 13, 14, 15, 16, -21, -11, 0 };
            foreach (int i in arr)
            {
                System.Console.Write("{0} \t " ,   i);
            }
        }
    }
}
```

Çıktı

```
11, 12, 13, 14, 15, 16, -21, -11, 0
```

Aşağıdaki program, bir metotla değer yazdırmakta, başka bir metotla değer okutmaktadır. Metotların parametreleri yaratılan array nesnedir.

Array.cs

```
using System;

namespace Arrayler
{
    class Array01
    {
        static int n = 10;
        static int[] arr = new int[n];

        static void değerYaz(int[] p)
        {
            for (int i = 0; i < p.Length; i++)
              p[i] = i;
        }

        static void değerOku(int[] r)
        {
            for (int i = 0; i < r.Length; i++)
                Console.WriteLine(r[i]);
        }
```

```
        static void Main()
        {
            değerYaz(arr);
            değerOku(arr);
        }
    }
}
```

Çıktı

```
0
1
2
3
4
5
6
7
8
9
```

Array Türleri

C# dilinde iki türlü array vardır. *Dikdörtgensel arrayler* ve *çentikli (jagged) arrayler.* Bunları örneklerle inceleyeceğiz.

Dikdörtgensel arrayler bir, iki ya da daha çok boyutlu olabilirler.

Bir Boyutlu Array

Bunlar tek indisli dizilerdir. 0-ıncı bileşenden başlayıp sonuncu bileşenine kadar index sırasıyla dizilirler. Yukarıda tanımladığımız arrayler ile aşağıdaki arrayler tek boyutludurlar.

```
int[]   derslik;
derslik = new int [3] {0, 1, 2};

string[]  kent = new string[5] {"Van", "Trabzon", "Kayseri", "Muş", "İzmir"};
string[]  kent = new string[] {"Van", "Trabzon", "Kayseri", "Muş", "İzmir"};
string[]  kent = {"Van", "Trabzon", "Kayseri", "Muş", "İzmir"};
```

Üçüncü satır, üç aşamalı array tanımını tek adımda yapmıştır. Dördüncü satır, dinamik uzunluklu array tanımıdır. Sonuncu satır, `new` kurucu deyimini kullanmadan array'i yaratmıştır.

`Length` arrayin uzunluğunu belirtir. Aşağıdaki program `Length` 'in işlevini göstermektedir.

Array01.cs

```
using System;

namespace Stringler
{
    class Array01
    {
        static void Main()
        {
            string[] kent = { "Van", "Trabzon", "Kayseri", "Muş",
```

```
"İzmir" };
            for (int i = 0; i < kent.Length; i++)
                Console.WriteLine("kent[{0}] = {1}", i, kent[i]);
        }
    }
}
```

Çıktı

```
kent[0] = Van
kent[1] = Trabzon
kent[2] = Kayseri
kent[3] = Muş
kent[4] = İzmir
```

Çok Boyutlu Array

Bileşenleri birden çok damgaya (index) bağlı olan arraylerdir. Aşağıdaki bildirimler, sırasıyla 1, 2 ve 3 boyutlu birer array bildirimidir.

```
int[,]   arr2Boyut ;       // 2 boyutlu array bildirimi
float[, ,]   arr3Boyut ;    // 3 boyutlu array bildirimi
float[, , ,]   arr4Boyut ;  // 4 boyutlu array bildirimi
```

Bir boyutlu arrayler için yaptığımız gibi, çok boyutlu arraylerin bileşenlerine de bildirim anında değer atayabiliriz. Aşağıdaki array, bileşenleri `int` tipi olan 2-boyutlu bir arraydir.

```
int[,] ikilSayı = new int[3, 2] { {1, 2}, {3, 4}, {5, 6} };
```

`ikilSayı` adlı arrayin bileşenlerini 3x2 tipi bir matris gibi yazdırabiliriz.

```
using System;

namespace Stringler
{
    class Array01
    {
        static void Main()
        {
            int[,] ikilSayı = new int[3, 2] { { 1, 2 }, { 3, 4 }, { 5,
6 } };
            Console.Write(ikilSayı[0, 0]); Console.Write("\t");
            Console.WriteLine(ikilSayı[0, 1]);
            Console.Write(ikilSayı[1, 0]); Console.Write("\t");
            Console.WriteLine(ikilSayı[1, 1]);
            Console.Write(ikilSayı[2, 0]); Console.Write("\t");
            Console.WriteLine(ikilSayı[2, 1]);


        }
    }
}
```

Çıktı

```
1       2
3       4
5       6
```

Arrayin bileşen sayısı matrisin bileşenlerinin sayısı kadardır. O halde, arrayin uzunluğu 3x2 = 6 dır; yani arrayin 6 bileşeni vardır. Bu matrise bakarak, arrayin bileşenlerini kolayca çıkarabiliriz.

Tek boyutlu arraylerin bileşenlerine olduğu gibi, çok boyutlu arraylerin bileşenlerine de seçkili (random) erişebiliriz. Örneğin, yukarıdaki programda

```
int[,] ikilSayı = new int[3, 2] { { 1, 2 }, { 3, 4 }, { 5, 6 } };
```

deyimi yerine şunları koyabiliriz:

```
int[,] ikilSayı;
ikilSayı = new int[3, 2];
        ikilSayı[0, 0] = 1;
        ikilSayı[0, 1] = 2;
        ikilSayı[1, 0] = 3;
        ikilSayı[1, 1] = 4;
        ikilSayı[2, 0] = 5;
        ikilSayı[2, 1] = 6;
```

Görüldüğü gibi her bileşen iki damga (index) ile belirleniyor. Bu nedenle, bu tür arraylere 2-boyutlu array diyoruz. Aşağıdaki örnek `string` tipinden iki boyutlu bir arraydir.

```
string[,] adSoyad = new string[2, 2] { {"Melih","Cevdet"}, {"Orhan","Veli"} };
```

Bunun bileşenlerini 2x2 türünden bir matris olarak yazabiliriz.

Melih	Cevdet
Orhan	Veli

Arrayin bileşenleri

```
adSoyad[0,0] = Melih
adSoyad[0,1] = Cevdet
adSoyad[1,0] = Orhan
adSoyad[1,1] = Veli
```

olur. Bu bileşenleri iç-içe döngü ile yazdırabiliriz.

Array.cs

```
using System;

namespace Stringler
{
    class Array01
    {
        static void Main()
        {
            string[,] adSoyad = new string[3, 2] { { "Melih", "Cevdet"
```

```
}, { "Orhan", "Veli" } , {"Oktay", "Rıfat"}};
            int i=0, j=0;
            for (i = 0; i < 3; i++)
           {
                for (j = 0; j < 2; j++)
                    Console.Write("adSoyad[{0},{1}] = {2} \t ", i, j,
adSoyad[i, j]);
                Console.WriteLine();
           }
       }
   }
}
```

Aşağıdaki deyimler eşdeğerdir:

```
int[,] ary = new int [2,3] { {1,2,3}, {4,5,6} };
int[,] ary = new int [,]   { {1,2,3}, {4,5,6} };
int[,] ary =               { {1,2,3}, {4,5,6} };
```

Array'lerin bileşenlerinin matris biçiminde yazılışı birer dikdörtgen görüntüsü veriyor. O nedenle, bazı kaynaklar bir ve birden çok boyutlu array'lere dikdörtgensel array derler. Dikdörtgensel olmayan arrayleri biraz sonra açıklayacağız.

Tek boyutlularda olduğu gibi, çok boyutlu arraylerin bileşenlerine de program koşarken değer atayabiliriz. Aşağıdaki program, iki boyutlu bir arrayin bileşenlerine iç-içe iki döngü ile değer atamaktadır.

Array.cs

```
using System;

namespace Arrayler
{
    class Array01
    {
        static void Main()
        {
            int[,] arr = new int[5, 4];
            for (int j = 0; j < 4; j++)      // satır döngüsü
           {
                for (int i = 0; i < 5; i++) //kolon döngüsü
               {
                    arr[i, j] = i*j ;       // bileşenlere değer atar
                    Console.Write(arr[i, j]);
                    Console.Write("\t");
               }
                Console.WriteLine();
           }
       }
   }
}
```

Çıktı

```
0   0   0   0   0
0   1   2   3   4
```

```
0    2    4    6    8
0    3    6    9    12
```

Aşağıdaki program iki boyutlu bir array üzerinde foreach döngüsünün yapılışını göstermektedir.

Array.cs

```
using System;

namespace Arrayler
{
    class Array01
    {
        static void Main()
        {
            int[,] arr2B = new int[3, 2] { { 3, 9 }, { 4, 16 }, { 5, 25 } };
             foreach (int i in arr2B)
            {
                System.Console.Write("{0} ", i);
            }
        }
    }
}
```

Çıktı

```
3       9       4       16      5       25
```

Array Arrayi (çentikli array)

Her veri tipinden array yapılabilir demiştik. Array de bir veri tipidir. Öyleyse, bileşenleri arrayler olan array tanımlanabilir. Matematikteki fonksiyon fonksiyonu kavramına benzer. Bu tür arraylere çentikli array *(jagged array)* de denilir. Her bileşeni bir array olduğu ve o arraylerin uzunlukları farklı olabileceği için, arrayin bileşenlerini bir kağıda satır satır yazarsak, ortaya çıkacak görüntü bir dikdörtgen değil, girintili çıkıntılı bir şekildir. Çentikli array denmesinin nedeni budur.

Bunu daha iyi açıklayabilmek için, bileşenleri, sırayla,

```
{1, 3, 5}
{2, 4, 6, 8, 10}
{111, 222}
```

olan bir `arAr` arrayini düşünelim. Bunu, alıştığımız biçimde yazarsak

```
arAr[0] = {1, 3, 5}
arAr[1] = {2, 4, 6, 8, 10}
arAr[2] = {111, 222}
```

olur. Bir adım daha gidelim ve bileşenleri { } parantezi içine koyalım ve ortaya çıkan arrayi `arAr[][]` ile gösterelim.

```
ArAr[][]  =  { arAr[0] ,  arAr[1] ,  arAr[2]  }
          =  { {1, 3, 5} ,  {2, 4, 6, 8, 10} ,  {111, 222} }
```

arAr[][] parantezlerinin ilki (soldaki) en dıştaki {} parantezinin içindeki bileşen sayısını belirtir. İkinci (sağdaki) [] ise içteki parantezlerin (bileşenlerin) bileşen sayılarını belirtir.

Bu notasyonda anlaştıktan sonra, array arrayinin bildirimini kolayca yapabiliriz.

```
int[]  a = new int[] { 1, 3, 5 };
int[]  b = new int[] { 2, 4, 6, 8, 10 };
int[]  c = new int[] { 111, 222 };
int[][] arAr = new int[][] { a, b, c };
```

Bunların ilk üçü, sırasıyla, içteki arrayleri yaratır. Sonuncu deyim ise, yaratılan üç arrayi kendi bileşenleri yapan array arrayini yaratır. İstersek dört adımda yapılan bu işi tek adıma indirebiliriz.

```
int[][] arAr = new int[][] { new int[] { 1, 3, 5 }, new int[] { 2, 4,
6, 8, 10 }, new int[] { 111, 222 } };
```

Buna göre çentikli arrayimizi yaratan ve bileşenlerini konsola gönderen programı şöyle yazabiliriz.

arAr01.cs

```
using System;

namespace Arrayler
{
    class ArAr01
    {
        static void Main()
        {
            int[] a = new int[] { 1, 3, 5 };
            int[] b = new int[] { 2, 4, 6, 8, 10 };
            int[] c = new int[] { 111, 222 };
            int[][] arAr = new int[][] { a, b, c };

            // Bu dört adım yerine aşağıdaki deyim konulabilir
            //int[][] arAr = new int[][] { new int[] { 1, 3, 5 }, new
int[] { 2, 4, 6, 8, 10 }, new int[] { 111, 222 } };

            for (int r = 0; r < arAr[0].Length; r++)
            {
                for (int i = 0; i < arAr[r].Length; i++)
                {
                    Console.Write(arAr[r][i]);
                    Console.Write("\t");
                }
                Console.WriteLine();
            }
        }
    }
```

Çıktı

```
1       3     5
2       4     6     8     10
111     222
```

Aşağıdaki program array arrayini farklı bir yöntemle tanımlamaktadır.

Array.cs

```
using System;

using System;
```

```
namespace Arrayler
{
    class Array04
    {
        static void Main()
        {
            int[][] j = new int[3][];
            j[0] = new int[] { 1, 2, 3 };
            j[1] = new int[] { 1, 2, 3, 4, 5, 6 };
            j[2] = new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

            for (int r = 0; r < 3; r++)
            {

                for (int i = 0; i < j[r].Length; i++)
                {
                    Console.Write(j[r][i]);
                    Console.Write("\t");
                }
                Console.WriteLine();
            }
        }
    }
}
```

Çıktı

```
1   2   3
1   2   3   4   5   6
1   2   3   4   5   6   7   8   9
```

Çıktıları görünce, array arraylerine neden çentikli dendiğini anlamış olmalısınız. Çentikli arraylerin bileşenleri bir tabloya dizildiğinde, dikdörtgensel bir biçim yerine, girintili çıkıntılı bir biçim almaktadırlar.

Alıştırmalar

Aşağıdaki deyimler, sırasıyla, tek boyutlu, çok boyutlu ve çentikli array'ler yaratır.

```
using System;


namespace Arrayler
{
    class TestArraysSınıfı
    {
        static void Main()
        {
            // Bir boyutlu array bildirimi
            int[] array1 = new int[5];

            // array'e bildirim anında değer atama
            int[] array2 = new int[] { 1, 3, 5, 7, 9 };

            // Alternatif sözdizimi
            int[] array3 = { 1, 2, 3, 4, 5, 6 };

            // 2 boyutlu array bildirimi
            int[,] multiDimensionalArray1 = new int[2, 3];
```

```
            // 2 boyutlu arraye bildirim anında değer atama
            int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 }
};

            // Çentikli array bildirimi
            int[][] çentikliArray = new int[6][];

            // ÇentikliArray'in ilk bileşenine atama
            çentikliArray[0] = new int[4] { 1, 2, 3, 4 };
        }
    }
}
```

Özet

Arrayler boyutlarına göre sınıflandırılırsa, bir array tek boyutlu ya da çok boyutlu olabilir.

Arrayler görünümlerine göre sınıflandırılırsa, bir array dikdörtgensel ya da çentikli olabilir.

Sayısal arraylerin öğelerinin öndeğerleri (default value) 0 dır. Referans öğelerin öndeğeri ise null'dır.

Çentikli array bileşenleri array olan arraydir. Dolayısıyla, bileşenleri referans tipindendir.

Arrayin bileşenleri 0 dan başlayarak numara sırası (indis, index) alır. İlk bileşeninin indisi 0 dır. n bileşenli bir arrayin sonuncu bileşeninin indisi (n-1) dir.

`[]` operatörü array'in bileşenlerini indisleriyle belirler.

Arrayin bileşenleri her veri tipinden olabilir. Array tipi de olabilir. Bunlara array arrayi ya da çentikli array denir.

Array tipleri referans tipindendir. Soyut Array tipinden yaratılmıştır. Ienumerable arayüzünü kullanır. Dolayısıyla, her array için foreach döngüsü kullanılabilir.

Array Sınıfı

Array yaratma, arraylerle işlem yapma, array içinde bileşen arama ve arrayi sıralama gibi arrayle ilgili işlemleri yapmaya yarayan öğeleri içeren bir sınıftır. Yedi tane özgeni (property) ve 30 dan fazla metodu vardır. Burada bir kaç örnek üzerinde duracağız:

Bir arrayin bileşen sayısını `Length` ya da `LongLength` özgenleri ile buluruz. Boyut sayısını `Rank` özgeni ile buluruz.

`Clear` metodu bir arrayin bileşen değerlerini siler. Sayısal bileşenleri `0`, referans tipleri `null` ve boolean tipleri `False` yapar.

`Clone` metodu bir arrayin kopyasını yapar.

`Copy` metodu bir arrayi aynı tipli başka bir array üzerine kopyalar.

`CreateInstance` metodu arrayin bir nesnesini yaratır.

`Equals` metodu arrayin iki nesnesinin eşit olup olmadığını söyler.

`Find` metodu array içinde aranan bir öğeyi bulur.

`GetValue` metodu arrayin istenen bir bileşeninin değerini bulur.

`IndexOf` metodu arrayin bir bileşeninin indisini bulur.

`Resize` metodu arrayin boyutunu değiştirir.

`Reverse` metodu tek boyutlu arrayin bileşenlerini ters sırada verir.

`SetValue` metodu bir bileşene değer atar.

`Sort` metodu bileşenleri sıralar

`ToString` metodu etkin nesneyi stringe dönüştürür.

Aşağıdaki program, Array sınıfının bazı metotlarının kullanılışını göstermektedir.

ArrayMetotları.cs

```
using System;
class ArrayMetotları
{
    static void Main()
    {
        // 5 bileşenli bir array yarat
        string[] adSoyad = new string[5];

        // 5 kişinin adını oku
        System.Console.WriteLine("Lütfen 5 kişinin adını giriniz:");
        for (int i = 0; i < adSoyad.Length; i++)
        {
            adSoyad[i] = System.Console.ReadLine();
        }

        // arrayin bileşenlerini giriş sırasıyla oku:
        System.Console.WriteLine("\nArray 'in orijinal sırası:");
        foreach (string ad in adSoyad)
        {
            System.Console.Write("{0}  ", ad);
        }

        // Arrayin sırasını tersine çevir:
        System.Array.Reverse(adSoyad);

        // Arrayi ters sırada yaz:
        System.Console.WriteLine("\n\nTers sıralı array:");
        foreach (string ad in adSoyad)
        {
            System.Console.Write("{0}  ", ad);
        }

        // Arrayi sırala (sort):
        System.Array.Sort(adSoyad);

        //  Sıralanmış arrayi yaz:
        System.Console.WriteLine("\n\n Sıralı Array:");
        foreach (string ad in adSoyad)
        {
            System.Console.Write("{0}  ", ad);
        }
    }
}
```

Çıktı

```
Lütfen 5 kişinin adını giriniz:
```

```
Hilâl
Esra
Ali
Güleda
Emrah

Array 'in orijinal sırası:
Hilâl  Esra  Ali  Güleda  Emrah

Ters sıralı array:
Emrah  Güleda  Ali  Esra  Hilâl

Sıralı Array:
Ali  Emrah  Esra  Güleda  Hilâl
```

Alıştırmalar

```
 using System;

class Uygulama
{
    static public void Main()
    {
        int[] number = { 1, 2, 3, 4, 5 };
        for (int i = 0; i < number.Length; i++) {
Console.WriteLine(number[i]); }
        for (int i = 0; i <= 4; i++) { Console.WriteLine(number[i]); }
        for (int i = 4; i >= 0; i--) { Console.WriteLine(number[i]); }
        foreach (int j in number) { Console.WriteLine(j); }
    }
}
```