Bölüm 23

Sıralama ve Arama

Sıralama ve Arama Kavramı

Sıralama ve arama (sorting/searching)) bilgisayar uygulamalarında çok önem taşır. Klâsik programlama dillerinde sıralama ve arama yapan kodları programcı kendisi yazardı. O nedenle, Bilgisayar uygulamalarında *Sıralama ve Arama Algoritmaları* denilen ayrı bir konu gelişmiştir. Yapılan işe göre en uygun algoritmayı seçmek programcının işiydi.

C# dili, programcının omuzundan bu yükü büyük ölçüde almıştır. Yalnız arrayleri değil, koleksiyonları sıralamak için kütüphanesine etkin yöntemler koymuştur. Özellikle metinlerin sıralanması alfabelere ve kültürlere bağlı olduğu için, `System.Globalization` aduzayı (namespace) sıralama işinde de devreye girer.

Aşağıda Türkçe alfabeye göre sıralama yapan bir program göreceksiniz.

Sıralama01.cs

```
using System;
using System.Globalization;
using System.Threading;
using System.Collections;
namespace Yöresellik
{
    class Sıralama01
    {
```

```
        static void Main(string[] args)
        {
            string[] kentler = {"Ünye", "ıhlara", "İzmir", "ğğğ" , "ĞĞĞ",
                                "Zonguldak", "Çanakkale", "Uşak",
                                "Kayseri", "Malatya", "çandır", "İstanbul",
                                "Ören", "Şanlıurfa", "Adana", "Afyon"};

            // arrayin sırasını değiştirmeden
            Console.WriteLine("\nSırasız...");
            adYaz(kentler);
            Console.WriteLine();

            // Türk alfabesine göre sıralanıyor
            Thread.CurrentThread.CurrentCulture =
               new CultureInfo("tr-TR");

            Array.Sort(kentler);
            Console.WriteLine("\nTürk Alfabesine göre Sıralı...");
            adYaz(kentler);

            // invariant kültüre göre
            Array.Sort(kentler, Comparer.DefaultInvariant);
            Console.WriteLine("\n DefaultInvariant kültüre göre
sıralama...");
            adYaz(kentler);
        }

        static void adYaz(IEnumerable e)
        {
            foreach (string s in e)
                Console.Write(s + " - ");
            Console.WriteLine();
        }
    }
}
```

Çıktı

```
Sırasız...
Ünye - ıhlara - İzmir - ğğğ - ĞĞĞ - Zonguldak - Çanakkale - Uşak - Kayseri - Mal
atya - çandır - İstanbul - Ören - Şanlıurfa - Adana - Afyon -

Türk Alfabesine göre Sıralı...
Adana - Afyon - Çanakkale - çandır - ğğğ - ĞĞĞ - ıhlara - İstanbul - İzmir - Kay
seri - Malatya - Ören - Şanlıurfa - Uşak - Ünye - Zonguldak -

DefaultInvariant kültüre göre sıralama...
Adana - Afyon - Çanakkale - çandır - ğğğ - ĞĞĞ - ıhlara - İstanbul - İzmir - Kay
seri - Malatya - Ören - Şanlıurfa - Ünye - Uşak - Zonguldak -
```

Türkçe Windows kullanıldığı için, *Türkçe Alfabe* 'ye göre sıralama ile *DefaultInvariant kültüre* göre sıralama aynı sonucu verir. İşletim Sistemi değişirse, bu iki sıralama farklı olabilir.

Aşağıdaki program `ArrayList` ile bir array yaratıyor, arrayi sıralıyor ve istenen bir bileşen değerinin hangi indisli bileşende olduğunu buluyor.

Sıralama02.cs

```
using System;
using System.Collections;

public class SortSearchDemo
{
    public static void Main()
    {
        // Bir ArrayList kur
        ArrayList arr = new ArrayList();

        // ArrayList'e bileşen ekle
        arr.Add(17);
        arr.Add(84);
        arr.Add(13);
        arr.Add(-46);
        arr.Add(3);
        arr.Add(-16);
        arr.Add(-8);
        arr.Add(9);
        arr.Add(21);
        arr.Add(-33);
        arr.Add(68);

        Console.Write("İlk ArrayList: ");
        foreach (int i in arr)
            Console.Write(i + " ");
        Console.WriteLine("\n");

        // Sırala
        arr.Sort();


        Console.Write("Sıralama sonrası ArrayList: ");
        foreach (int i in arr)
            Console.Write(i + " ");
        Console.WriteLine("\n");

        Console.WriteLine("-33 'ün indeksi = " +
                          arr.BinarySearch(-33));
    }
}
```

Çıktı

```
İlk ArrayList: 17 84 13 -46 3 -16 -8 9 21 -33 68
Sıralama sonrası ArrayList :  -46 -33 -16 -8 3 9 13 17 21 68 84
-33 'ün indeksi = 1
```

Ters Sıralama

Bazı durumlarda arrayi ters (azalan) yönde sıralamamız gerekir. Bunu yapan etkin yöntemler geliştirilebilir; ama boyu çok uzun olmayan arraylerde, artan yönde sıraladıktan sonra `Reverse()` metodunu kullanmak basitlik sağlar. Ancak bu yöntem çok büyük arraylerde bellek kullanımı ve zaman açısından yeğlenmez.

Sıralama03.cs

```
using System;
using System.Collections;
```

```
using System.Collections.Generic;
using System.Text;

class Program
{
    static void Main(string[] args)
    {
        ArrayList futbolTakımı = new ArrayList();
        futbolTakımı.Add("Fenerbahçe");
        futbolTakımı.Add("Galatasaray");
        futbolTakımı.Add("Beşiktaş");
        futbolTakımı.Add("Trabzon");

        Console.Write("Sıralamadan önce : \n");

        foreach (string takım in futbolTakımı)
        {
            Console.Write("\n" + takım + "\n");
        }
        futbolTakımı.Sort();
        futbolTakımı.Reverse();
        Console.Write("\n Sıralamadan sonra : \n");
        foreach (string takım in futbolTakımı)
        {
            Console.Write("\n" + takım + "\n");
        }
    }
}
```

Çıktı

```
Sıralamadan önce :

Fenerbahçe
Galatasaray
Beşiktaş
Trabzon

 Sıralamadan sonra :
Trabzon
Galatasaray
Fenerbahçe
Beşiktaş
```

Aynı yöntemi sayısal diziler için de kullanabiliriz.

Sıralama04.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Text;

class Program
{
    static void Main(string[] args)
    {
        int[] dizi = new int[] { 3, 1, 4, 5, 2 };
```

```
        Array.Sort(dizi);
        Array.Reverse(dizi);

        Console.Write("Sıralamadan önce : \n");

        for (int i=0 ; i<dizi.Length ; i++)
        {
            Console.WriteLine(dizi[i]);
        }
    }
}
```

`Icomparable` arayüzü kullanılarak, ters sıralamayı yapan bir program yazılabilir. Aşağıda bir örnek görülmektedir.

Sıralama05.cs

```
using System;
using System.Collections.Generic;

namespace CSharp411
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] array = new int[] { 3, 1, 4, 5, 2 };
            ArraySorter<int>.SortDescending(array);
            WriteArray(array);
            Console.ReadLine();
        }
        static private void WriteArray(int[] array)
        {
            foreach (int i in array)
            {
                Console.WriteLine(i);
            }
        }
    }
    static public class ArraySorter<T>
        where T : IComparable
    {
        static public void SortDescending(T[] array)
        {
            Array.Sort<T>(array, s_Comparer);
        }
        static private ReverseComparer s_Comparer = new
ReverseComparer();
        private class ReverseComparer : IComparer<T>
        {
            public int Compare(T object1, T object2)
            {
                return -((IComparable)object1).CompareTo(object2);
            }
        }
    }
}
```

Arama Teknikleri

Bilgisayar programlarında sıralama gibi önemli olan öteki konu arama eylemidir. *Arama Algoritmaları* (Search Algorithms) programlamanın önemli araştırma alanlarından birisi olagelmiştir. Bu konuda çok etkili teknikler geliştirilmiştir. Çok kullanılanlardan birisi *binary search*, ötekisi *linear search* (doğrusal arama) tekniğidir. Doğrusal aramada, aranan değer arrayin başından sonuna doğru her öğesi ile karşılaştırılır. Bulduğunda arama durur. Bulamazsa sonuna kadar devam eder. *Binary search* tekniğinde ise, yarılama denilen bir yöntem kullanılır. Aranan öğe, arrayin başlarında ise doğrusal arama daha çabuk bulur. Değilse, binary search daha çabuk bulur. Aranan öğenin nerede olduğu bilinmediğine göre binary search tekniği tercih edilir.

Doğrusal Arama

Aşağıdaki program, kullanıcıdan aldığı verilerle bir tamsayı array oluşturuyor, sonra kullanıcının istediği bir veriyi array içinde arıyor. Bulursa, onun arrayin kaçıncı öğesi olduğunu , bulamazsa o sayının array içinde olmadığını konsola yazıyor. Kodların açıklamaları kaynak programa yazıldığı için, çözümlemeyi vermeye gerek yoktur.

DoğrusalArama01.cs

```
using System;

class Arama
{
    public static void DoğrusalArama()
    {
        //tamsayılardan oluşan bir array kur
        int[] numArray = new int[100];
        //Girilen veri sayı değilse
        bool isNum = false;
        //array'in boyutu
        int sizeNum;
        //kullanıcıdan arrayin boyutunu sor
        Console.WriteLine("Arrayin boyutunu giriniz");
        //Girilen sayıyı oku
        string sizeString = Console.ReadLine();
        //TryParse kullanarak girilen verinin sayısal olduğunu denetle
        isNum = Int32.TryParse(sizeString, out sizeNum);
        // true/false
        if (isNum)
        {
            //Arrayin öğelerini gir
            Console.WriteLine("-------------------------");
            Console.WriteLine("\n Arrayin sayısal öğelerini gir  \n");
            // girilen sayıları bir döngü ile oku
            for (int i = 0; i < sizeNum; i++)
            {
                int tempNum;
                // her bileşeni ayrı ayrı Enter ile gir
                string elements = Console.ReadLine();
                //TryParse ile girilen verinin sayısal olduğunu denetle
                isNum = Int32.TryParse(elements, out tempNum);
                if (isNum)
                {
                    //girilen veri sayısal ise, arraye ata
                    numArray[i] = tempNum;
                }
                else
                {
                    Console.WriteLine("Sayısal olmayan veri girdiniz!");
```

```
                    break;
                }
            }
            //Kullanıcıdan arayacağı sayıyı iste
            Console.WriteLine("-------------------------");
            Console.WriteLine("Aranacak sayıyı giriniz \n");
            //gririlen veriyi oku
            string searchString = Console.ReadLine();
            //girilecek sayıyı tutacak değişken
            int searchNum;
            isNum = Int32.TryParse(searchString, out searchNum);
            //Girilen veri sayısal mı?
            if (isNum)
            {
                //Arrayi döngü ile tara
                for (int i = 0; i < sizeNum; i++)
                {
                    //Eğer aranan değer, array içinde varsa
                    //kullanıcıya hangi indis olduğunu bildir.
                    if (numArray[i] == searchNum)
                    {
                        Console.WriteLine("-------------------------");
                        Console.WriteLine("Aradığınız {0} sayısı dizinin
{1} -inci öğesidir \n", searchNum, i + 1);
                        return;
                    }
                }
                //Aranan sayı bulunmamışsa
                Console.WriteLine("Aranan {0} sayısı bulunamadı \n",
searchNum);
            }
            else
            {
                //Kullanıcı aranacak veriyi sayısal girmediyse
                Console.WriteLine("Aranmak için sayısal bir değer
girmediniz!");
            }
        }
        else
        {
            Console.WriteLine("Lütfen bir sayı giriniz!");
        }
    }
    static void Main(string[] args)
    {
        DoğrusalArama();
    }
}
```

Binary Arama

Aşağıdaki program, verilen bir sayının sıralı bir array içinde olup olmadığını binary search yöntemiyle aramaktadır.

BinaryArama01.cs

```
using System;
```

```
class ArrayBinarySearch
{
    public static void Main()
    {
        int[] arr = { 10, 20, 30, 40, 50, 1000 };
        Console.WriteLine("Arrayin öğeleri: ");
        for (int i = 0; i < arr.Length; i++)
        {
            Console.Write("arr[{0}] = {1, -5}", i, arr[i]);
        }
        Console.WriteLine();
        FindObject(arr, 20);
        FindObject(arr, 35);
        FindObject(arr, 10000);
    }

    public static void FindObject(Array ary, Object o)
    {
        int index = Array.BinarySearch(ary, 0, ary.Length, o);
        Console.WriteLine();
        if (index > 0)
        {
            Console.WriteLine("Object: {0} -nin indisi :  [{1}]", o,
index);
        }
        else if (index == ary.Length)
        {
            Console.WriteLine("Object: {0} bulunamadı. "
               + "Arama bitti.", o);
            Console.WriteLine();
        }
        else
        {
            Console.WriteLine("Object: {0} bulunamadı. "
               + "Aramanın eriştiği indis : [{1}].", o, ~index-1);
        }
    }
}
```

Çıktı

```
Arrayin öğeleri:
arr[0] = 10   arr[1] = 20   arr[2] = 30   arr[3] = 40   arr[4] = 50   arr[5] = 1000
Object: 20 -nin indisi :  [1
Object: 35 bulunamadı. Aramanın eriştiği indis : [2]
Object: 10000 bulunamadı. Aramanın eriştiği indis : [5].
```

Programı çözümleme:

`ArrayBinarySearch` adlı sınıf içinde aramayı yapmak üzere

```
public static void FindObject(Array ary, Object o)
```

metodu tanımlanıyor. Bu metot Array sınıfının

```
BinarySearch(ary, 0, ary.Length, o)
```

metodunu kullanıyor. Bu metodun aşkın sürümleri var. Program aranan öğe bulunursa, bulunduğu bileşenin indisini yazıyor, bulamazsa, eriştiği bileşenin indisini yazıyor. Array sıralı olduğu için bunu yapması kolaydır. Aranan öğeden büyük değerli bir bileşene erişirse, aranan öğenin array içinde olmayacağı açıktır.