Άσκηση 1: Βασική Υλοποίηση του Quick Sort

Περιγραφή: Υλοποίησε τον αλγόριθμο Quick Sort για να ταξινομήσεις έναν πίνακα ακεραίων.

Κώδικας:

```
using System;

class Program
{
    static void Main()
    {
        int[] array = { 34, 7, 23, 32, 5, 62 };
        QuickSort(array, 0, array.Length - 1);

        Console.WriteLine("Ταξινομημένος Πίνακας:");
        foreach (var item in array)
        {
            Console.Write(item + " ");
        }
    }

    static void QuickSort(int[] array, int low, int high)
    {
        if (low < high)
        {
            int pivotIndex = Partition(array, low, high);
            QuickSort(array, low, pivotIndex - 1);
            QuickSort(array, pivotIndex + 1, high);
        }
    }

    static int Partition(int[] array, int low, int high)
    {
        int pivot = array[high];
        int i = low - 1;

        for (int j = low; j < high; j++)
        {
            if (array[j] <= pivot)
            {
                i++;
                Swap(ref array[i], ref array[j]);
            }
        }
        Swap(ref array[i + 1], ref array[high]);
        return i + 1;
```

```
    }

    static void Swap(ref int a, ref int b)
    {
        int temp = a;
        a = b;
        b = temp;
    }
}
```

Άσκηση 2: Προσαρμογή για Αντίστροφη Ταξινόμηση

Περιγραφή: Τροποποίησε την υλοποίηση του Quick Sort ώστε να ταξινομεί τον πίνακα σε φθίνουσα σειρά.

Κώδικας:

```
using System;

class Program
{
    static void Main()
    {
        int[] array = { 34, 7, 23, 32, 5, 62 };
        QuickSort(array, 0, array.Length - 1);

        Console.WriteLine("Ταξινομημένος Πίνακας (Φθίνουσα Σειρά):");
        foreach (var item in array)
        {
            Console.Write(item + " ");
        }
    }

    static void QuickSort(int[] array, int low, int high)
    {
        if (low < high)
        {
            int pivotIndex = Partition(array, low, high);
            QuickSort(array, low, pivotIndex - 1);
            QuickSort(array, pivotIndex + 1, high);
        }
    }
```

```
    static int Partition(int[] array, int low, int high)
    {
        int pivot = array[high];
        int i = low - 1;

        for (int j = low; j < high; j++)
        {
            if (array[j] >= pivot) // Αλλαγή σύγκρισης για φθίνουσα σειρά
            {
                i++;
                Swap(ref array[i], ref array[j]);
            }
        }
        Swap(ref array[i + 1], ref array[high]);
        return i + 1;
    }

    static void Swap(ref int a, ref int b)
    {
        int temp = a;
        a = b;
        b = temp;
    }
}
```

Άσκηση 3: Εφαρμογή Quick Sort για Πίνακα Αλφαριθμητικών

Περιγραφή: Υλοποίησε τον αλγόριθμο Quick Sort για να ταξινομήσεις έναν πίνακα από αλφαριθμητικά (strings).

Κώδικας:

```
using System;

class Program
{
    static void Main()
    {
        string[] array = { "apple", "orange", "banana", "pear", "grape" };
        QuickSort(array, 0, array.Length - 1);

        Console.WriteLine("Ταξινομημένος Πίνακας:");
        foreach (var item in array)
        {
            Console.Write(item + " ");
        }
```

```
    }

    static void QuickSort(string[] array, int low, int high)
    {
        if (low < high)
        {
            int pivotIndex = Partition(array, low, high);
            QuickSort(array, low, pivotIndex - 1);
            QuickSort(array, pivotIndex + 1, high);
        }
    }

    static int Partition(string[] array, int low, int high)
    {
        string pivot = array[high];
        int i = low - 1;

        for (int j = low; j < high; j++)
        {
            if (string.Compare(array[j], pivot) <= 0)
            {
                i++;
                Swap(ref array[i], ref array[j]);
            }
        }
        Swap(ref array[i + 1], ref array[high]);
        return i + 1;
    }

    static void Swap(ref string a, ref string b)
    {
        string temp = a;
        a = b;
        b = temp;
    }
}
```

Άσκηση 4: Μέτρηση Χρόνου Εκτέλεσης

Περιγραφή: Τροποποίησε την υλοποίηση του Quick Sort για να μετράει και να εμφανίζει τον χρόνο εκτέλεσης της ταξινόμησης.

Κώδικας:

```
using System;
using System.Diagnostics;

class Program
{
    static void Main()
    {
        int[] array = { 34, 7, 23, 32, 5, 62 };
        Stopwatch stopwatch = new Stopwatch();

        stopwatch.Start();
        QuickSort(array, 0, array.Length - 1);
        stopwatch.Stop();

        Console.WriteLine("Ταξινομημένος Πίνακας:");
        foreach (var item in array)
        {
            Console.Write(item + " ");
        }

        Console.WriteLine($"\nΧρόνος Εκτέλεσης: {stopwatch.ElapsedMilliseconds} ms");
    }

    static void QuickSort(int[] array, int low, int high)
    {
        if (low < high)
        {
            int pivotIndex = Partition(array, low, high);
            QuickSort(array, low, pivotIndex - 1);
            QuickSort(array, pivotIndex + 1, high);
        }
    }

    static int Partition(int[] array, int low, int high)
    {
        int pivot = array[high];
        int i = low - 1;

        for (int j = low; j < high; j++)
        {
            if (array[j] <= pivot)
            {
                i++;
                Swap(ref array[i], ref array[j]);
            }
        }
```

```
        Swap(ref array[i + 1], ref array[high]);
        return i + 1;
    }

    static void Swap(ref int a, ref int b)
    {
        int temp = a;
        a = b;
        b = temp;
    }
}
```

Άσκηση 5: Χρήση του Quick Sort σε Γενικότερη Κλάση

Περιγραφή: Δημιούργησε μια γενικότερη κλάση που να υλοποιεί τον αλγόριθμο Quick Sort ώστε να μπορεί να χρησιμοποιηθεί για οποιοδήποτε τύπο δεδομένων που υλοποιεί τη διεπαφή `IComparable`.

Κώδικας:

```
using System;

class QuickSorter<T> where T : IComparable<T>
{
    public static void QuickSort(T[] array)
    {
        QuickSort(array, 0, array.Length - 1);
    }

    private static void QuickSort(T[] array, int low, int high)
    {
        if (low < high)
        {
            int pivotIndex = Partition(array, low, high);
            QuickSort(array, low, pivotIndex - 1);
            QuickSort(array, pivotIndex + 1, high);
        }
    }

    private static int Partition(T[] array, int low, int high)
    {
        T pivot = array[high];
        int i = low - 1;
```

```
        for (int j = low; j < high; j++)
        {
            if (array[j].CompareTo(pivot) <= 0)
            {
                i++;
                Swap(ref array[i], ref array[j]);
            }
        }
        Swap(ref array[i + 1], ref array[high]);
        return i + 1;
    }

    private static void Swap(ref T a, ref T b)
    {
        T temp = a;
        a = b;
        b = temp;
    }
}

class Program
{
    static void Main()
    {
        int[] intArray = { 34, 7, 23, 32, 5, 62 };
        string[] stringArray = { "apple", "orange", "banana", "pear", "grape" };

        QuickSorter<int>.QuickSort(intArray);
        QuickSorter<string>.QuickSort(stringArray);

        Console.WriteLine("Ταξινομημένος Πίνακας Ακεραίων:");
        foreach (var item in intArray)
        {
            Console.Write(item + " ");
        }

        Console.WriteLine("\nΤαξινομημένος Πίνακας Αλφαριθμητικών:");
        foreach (var item in stringArray)
        {
            Console.Write(item + " ");
        }
    }
}
```