Άσκηση 1: Απλή Ταξινόμηση Ακέραιων Αριθμών

Περιγραφή: Δημιούργησε μια εφαρμογή C# που χρησιμοποιεί τον αλγόριθμο Radix Sort για να ταξινομήσει έναν πίνακα από θετικούς ακέραιους αριθμούς.

Κώδικας:

```
using System;

class RadixSortExercise1
{
    static void Main()
    {
        int[] numbers = { 170, 45, 75, 90, 802, 24, 2, 66 };
        RadixSort(numbers);

        Console.WriteLine("Ταξινομημένοι αριθμοί:");
        foreach (var num in numbers)
        {
            Console.Write(num + " ");
        }
    }

    static void RadixSort(int[] array)
    {
        int max = GetMax(array);
        for (int exp = 1; max / exp > 0; exp *= 10)
        {
            CountingSort(array, exp);
        }
    }

    static void CountingSort(int[] array, int exp)
    {
        int n = array.Length;
        int[] output = new int[n];
        int[] count = new int[10];

        // Αρχικοποιούμε τον πίνακα count
        Array.Fill(count, 0);
```

```
        // Μετράμε την εμφάνιση κάθε ψηφίου
        for (int i = 0; i < n; i++)
        {
            int digit = (array[i] / exp) % 10;
            count[digit]++;
        }

        // Αθροίζουμε τα στοιχεία του πίνακα count
        for (int i = 1; i < 10; i++)
        {
            count[i] += count[i - 1];
        }

        // Δημιουργούμε το output array
        for (int i = n - 1; i >= 0; i--)
        {
            int digit = (array[i] / exp) % 10;
            output[count[digit] - 1] = array[i];
            count[digit]--;
        }

        // Αντιγράφουμε τα στοιχεία στο αρχικό array
        Array.Copy(output, array, n);
    }

    static int GetMax(int[] array)
    {
        int max = array[0];
        foreach (int num in array)
        {
            if (num > max)
            {
                max = num;
            }
        }
        return max;
    }
}
```

Λύση: Ο κώδικας ταξινομεί έναν πίνακα από ακέραιους αριθμούς χρησιμοποιώντας τον

αλγόριθμο Radix Sort. Αρχικά υπολογίζει τον μέγιστο αριθμό για να καθορίσει πόσες ψηφιακές θέσεις χρειάζεται να εξετάσει. Στη συνέχεια, χρησιμοποιεί την Counting Sort για κάθε ψηφιακή θέση (εκατοντάδες, δεκάδες, μονάδες) για να ταξινομήσει τον πίνακα.

Άσκηση 2: Ταξινόμηση Μεγάλων Αριθμών

Περιγραφή: Δημιούργησε μια εφαρμογή C# που χρησιμοποιεί τον αλγόριθμο Radix Sort για να ταξινομήσει έναν πίνακα από μεγάλα ακέραια νούμερα.

Κώδικας:

```
using System;

class RadixSortExercise2
{
    static void Main()
    {
        int[] numbers = { 123456, 987654, 564738, 123789, 789456, 456123 };
        RadixSort(numbers);

        Console.WriteLine("Ταξινομημένοι αριθμοί:");
        foreach (var num in numbers)
        {
            Console.Write(num + " ");
        }
    }

    static void RadixSort(int[] array)
    {
        int max = GetMax(array);
        for (int exp = 1; max / exp > 0; exp *= 10)
        {
            CountingSort(array, exp);
        }
```

```
}

static void CountingSort(int[] array, int exp)
{
    int n = array.Length;
    int[] output = new int[n];
    int[] count = new int[10];

    // Αρχικοποιούμε τον πίνακα count
    Array.Fill(count, 0);

    // Μετράμε την εμφάνιση κάθε ψηφίου
    for (int i = 0; i < n; i++)
    {
        int digit = (array[i] / exp) % 10;
        count[digit]++;
    }

    // Αθροίζουμε τα στοιχεία του πίνακα count
    for (int i = 1; i < 10; i++)
    {
        count[i] += count[i - 1];
    }

    // Δημιουργούμε το output array
    for (int i = n - 1; i >= 0; i--)
    {
        int digit = (array[i] / exp) % 10;
        output[count[digit] - 1] = array[i];
        count[digit]--;
    }

    // Αντιγράφουμε τα στοιχεία στο αρχικό array
    Array.Copy(output, array, n);
}

static int GetMax(int[] array)
{
    int max = array[0];
    foreach (int num in array)
```

```
        {
            if (num > max)
            {
                max = num;
            }
        }
        return max;
    }
}
```

Λύση: Ο κώδικας χρησιμοποιεί την ίδια λογική με την προηγούμενη άσκηση αλλά με μεγαλύτερους αριθμούς. Ο Radix Sort εξακολουθεί να ταξινομεί τους αριθμούς μέσω των ψηφίων τους.

Άσκηση 3: Ταξινόμηση Αριθμών Με Διαφορετικά Σωστά Ψηφία

Περιγραφή: Δημιούργησε μια εφαρμογή C# που χρησιμοποιεί τον αλγόριθμο Radix Sort για να ταξινομήσει έναν πίνακα από αριθμούς που περιέχουν διαφορετικό αριθμό ψηφίων.

Κώδικας:

```
using System;

class RadixSortExercise3
{
    static void Main()
    {
        int[] numbers = { 7, 45, 123, 567, 89, 234, 1, 12 };
        RadixSort(numbers);

        Console.WriteLine("Ταξινομημένοι αριθμοί:");
        foreach (var num in numbers)
        {
            Console.Write(num + " ");
        }
    }
```

```
static void RadixSort(int[] array)
{
    int max = GetMax(array);
    for (int exp = 1; max / exp > 0; exp *= 10)
    {
        CountingSort(array, exp);
    }
}

static void CountingSort(int[] array, int exp)
{
    int n = array.Length;
    int[] output = new int[n];
    int[] count = new int[10];

    // Αρχικοποιούμε τον πίνακα count
    Array.Fill(count, 0);

    // Μετράμε την εμφάνιση κάθε ψηφίου
    for (int i = 0; i < n; i++)
    {
        int digit = (array[i] / exp) % 10;
        count[digit]++;
    }

    // Αθροίζουμε τα στοιχεία του πίνακα count
    for (int i = 1; i < 10; i++)
    {
        count[i] += count[i - 1];
    }

    // Δημιουργούμε το output array
    for (int i = n - 1; i >= 0; i--)
    {
        int digit = (array[i] / exp) % 10;
        output[count[digit] - 1] = array[i];
        count[digit]--;
    }
```

```
        // Αντιγράφουμε τα στοιχεία στο αρχικό array
        Array.Copy(output, array, n);
    }

    static int GetMax(int[] array)
    {
        int max = array[0];
        foreach (int num in array)
        {
            if (num > max)
            {
                max = num;
            }
        }
        return max;
    }
}
```

Λύση: Ο κώδικας χρησιμοποιεί τον αλγόριθμο Radix Sort για να ταξινομήσει αριθμούς με διαφορετικό αριθμό ψηφίων, διασφαλίζοντας ότι όλοι οι αριθμοί ταξινομούνται σωστά, ανεξάρτητα από το μήκος τους.