Лабораторна робота №3

by Тукало Іван

11 варіант

Завдання (блок) 1:

Вставити в початок масиву мінімум з усіх значень масиву, а в кінець — максимум.

Слід зробити «максимально вручну», без складних бібліотечних методів (дозволені лише new, Length, квадратні дужки, і, можливо, Resize). Зокрема, прямо заборонено вживати в цьому блоці Array.ConvertAll, а також складну роботу з рядками де б не було, крім безпосередньо введення та виведення.

Якщо вже зроблено цими «максимально ручними» засобами, то, як додаткове завдання на додаткові бали можна зробити ще раз, максимально користуючись бібліотечними методами. Також на додаткові бали можна зробити ще раз, використавши замість масиву колекцію List і максимально користуючись методами колекції List, але теж лише після того, як зроблено масивами «максимально вручну».

Для цього завдання слід зробити і код (текст) програми, і блок-схеми (такі, як у 1-му семестрі; блок-схеми (багато) — в тому смислі, що для кожного методу окремо), і діаграми посилань (аналогічні наведеним у tutorial_dyn_arrays.pdf,

Завдання (блок) 2, без варіантів:

2а) Прочитайте єдине натуральне число n, спочатку сформуйте у пам'яті у вигляді зубчастого масиву з рядками різної довжини, потім виведіть на екран такі переліки: для кожного і від 0 до n–1, послідовність номер і містить, у порядку зростання, ті й тільки ті числа від 1 до n, які кратні сумі цифр числа і.

2б) Результат виведення на екран повинен бути абсолютно таким самим, як у попередньому пункті, але в пам'яті повинно бути значно менше різних «вкладених» одновимірних масивів, за рахунок того, що різні числа можуть мати однакову суму цифр, у цих випадках виходить однакова послідовність, і її можна тримати в пам'яті один раз.

Проведіть вимірювання технічними засобами обсягів використаної пам'яті в кожному з цих варіантів і проаналізуйте це своїми словами.

Для цього завдання слід зробити лише код (текст) програми та щойно згаданий аналіз. Ні блок-схем, ні діаграм посилань не треба.

Завдання (блок) 3:

Додати рядок після рядка, що містить максимальний елемент (якщо у різних місцях є кілька елементів з однаковим максимальним значенням, то брати перший з них).

На додаткові бали можна зробити також із використанням замість масиву колекції List; але зробити через зубчастий масив (звичайний масив звичайних масивів) теж треба, і спосіб з List-ом перевірятиметься й оцінюватиметься лише якщо зроблено через звичайний масив звичайних масивів.

Зробити діаграми посилань

Завдання (блок) 4:

Переписати з кожного рядка матриці P у відповідні рядки матриці Q лише непарні елементи. Відсортувати рядки отриманої матриці Q за зростанням. Сформувати з рядків матриці Q одновимірний масив та визначити ті його елементи, значення яких співпадають із власним індексом.

КОД:

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

class Program
{
    static void DoBlock_1()
    {
        int[] array;

        Console.Write("Бажаєте випадкове заповнення масиву? (так[1]/ні[2]): ");
        string userInput = Console.ReadLine().ToLower();

        if (userInput == "1")
        {
            array = GenerateRandomArray();
        }
        else
        {
            Console.Write("Бажаєте вводити кожен елемент у окремому рядку? (так[1]/ні[2]): ");
            userInput = Console.ReadLine().ToLower();

            if (userInput == "1")
            {
                array = InputArraySeparateLines();
            }
            else
            {
                array = InputArraySingleLine();
            }
        }

        Console.WriteLine("\nВаш масив:");
        PrintArray(array);

        TransformArray(ref array);

        Console.WriteLine("\nМасив після вставки мінімуму в початок та максимуму в кінець:");
        PrintArray(array);
        Console.WriteLine();
    }
    static void TransformArray(ref int[] array)
    {
        if (array.Length == 0) return;

        int minValue = array[0];
        int maxValue = array[0];

        for (int i = 1; i < array.Length; i++)
        {
            if (array[i] < minValue) minValue = array[i];
            if (array[i] > maxValue) maxValue = array[i];
        }

        Array.Resize(ref array, array.Length + 2);

        // Зсуваємо всі елементи на одну позицію вправо для вставки мінімуму на початок
        for (int i = array.Length - 1; i > 0; i--)
        {
            array[i] = array[i - 1];
        }

        array[0] = minValue;
        array[array.Length - 1] = maxValue;
    }
    static int[] GenerateRandomArray()
    {
        Console.Write("Введіть кількість елементів масиву: ");
        int length = int.Parse(Console.ReadLine());

        Random random = new Random();
        int[] array = new int[length];
        Console.Write("Введіть мінімальне число масиву: ");
        int min = int.Parse(Console.ReadLine());
        Console.Write("Введіть максимальне число масиву: ");
        int max = int.Parse(Console.ReadLine());

        for (int i = 0; i < length; i++)
        {
            array[i] = random.Next(min, max);
        }

        return array;
```

```
}

static int[] InputArraySeparateLines()
{
    Console.Write("Введіть кількість елементів масиву: ");
    int length = int.Parse(Console.ReadLine());

    int[] array = new int[length];

    Console.WriteLine("Введіть елементи масиву по одному у кожному рядку:");
    for (int i = 0; i < length; i++)
    {
        Console.Write($"Елемент {i + 1}: ");
        array[i] = int.Parse(Console.ReadLine());
    }

    return array;
}

static int[] InputArraySingleLine()
{
    Console.Write("Введіть елементи масиву, розділені пробілами або табуляціями: ");
    string[] input = Console.ReadLine().Split(new char[] { ' ', '\t' }, StringSplitOptions.RemoveEmptyEntries);

    int[] array = new int[input.Length];

    for (int i = 0; i < input.Length; i++)
    {
        array[i] = int.Parse(input[i]);
    }

    return array;
}
static void PrintArray(int[] array)
{
    Console.WriteLine(string.Join(", ", array));
}
static void DoBlock_5()
{
    int[] array;

    Console.Write("Бажаєте випадкове заповнення масиву? (так[1]/ні[2]): ");
    string userInput = Console.ReadLine().ToLower();

    if (userInput == "1")
    {
        array = GenerateRandomArray1();
    }
    else
    {
        Console.Write("Бажаєте вводити кожен елемент у окремому рядку? (так[1]/ні[2]): ");
        userInput = Console.ReadLine().ToLower();

        if (userInput == "1")
        {
            array = InputArraySeparateLines1();
        }
        else
        {
            array = InputArraySingleLine1();
        }
    }

    Console.WriteLine("\nВаш масив:");
    PrintArray1(array);

    TransformArray1(ref array);

    Console.WriteLine("\nМасив після вставки мінімуму в початок та максимуму в кінець:");
    PrintArray1(array);
    Console.WriteLine();
}

static void TransformArray1(ref int[] array)
{
    if (array.Length == 0) return;

    int minValue = array.Min();
    int maxValue = array.Max();

    array = new int[] { minValue }.Concat(array).Concat(new int[] { maxValue }).ToArray();
}

static int[] GenerateRandomArray1()
{
    Console.Write("Введіть кількість елементів масиву: ");
    int length = int.Parse(Console.ReadLine());
```

```
    Random random = new Random();
    int[] array = new int[length];
    Console.Write("Введіть мінімальне число масиву: ");
    int min = int.Parse(Console.ReadLine());
    Console.Write("Введіть максимальне число масиву: ");
    int max = int.Parse(Console.ReadLine());

    for (int i = 0; i < length; i++)
    {
        array[i] = random.Next(min, max);
    }

    return array;
}

static int[] InputArraySeparateLines1()
{
    Console.Write("Введіть кількість елементів масиву: ");
    int length = int.Parse(Console.ReadLine());

    int[] array = new int[length];

    Console.WriteLine("Введіть елементи масиву по одному у кожному рядку:");
    for (int i = 0; i < length; i++)
    {
        Console.Write($"Елемент {i + 1}: ");
        array[i] = int.Parse(Console.ReadLine());
    }

    return array;
}

static int[] InputArraySingleLine1()
{
    Console.Write("Введіть елементи масиву, розділені пробілами або табуляціями: ");
    string[] input = Console.ReadLine().Split(new char[] { ' ', '\t' }, StringSplitOptions.RemoveEmptyEntries);

    int[] array = new int[input.Length];

    for (int i = 0; i < input.Length; i++)
    {
        array[i] = int.Parse(input[i]);
    }

    return array;
}

static void PrintArray1(int[] array)
{
    Console.WriteLine(string.Join(", ", array));
}
static void DoBlock_6()
{
    List<int> list;

    Console.Write("Бажаєте випадкове заповнення масиву? (так[1]/ні[2]): ");
    string userInput = Console.ReadLine().ToLower();

    if (userInput == "1")
    {
        list = GenerateRandomList();
    }
    else
    {
        Console.Write("Бажаєте вводити кожен елемент у окремому рядку? (так[1]/ні[2]): ");
        userInput = Console.ReadLine().ToLower();

        if (userInput == "1")
        {
            list = InputListSeparateLines();
        }
        else
        {
            list = InputListSingleLine();
        }
    }

    Console.WriteLine("\nВаш масив:");
    PrintList(list);

    TransformList(list);

    Console.WriteLine("\nМасив після вставки мінімуму в початок та максимуму в кінець:");
    PrintList(list);
    Console.WriteLine();
}

static void TransformList(List<int> list)
```

```
{
    if (list.Count == 0) return;

    int minValue = list.Min();
    int maxValue = list.Max();

    list.Insert(0, minValue);
    list.Add(maxValue);
}

static List<int> GenerateRandomList()
{
    Console.Write("Введіть кількість елементів масиву: ");
    int length = int.Parse(Console.ReadLine());

    Random random = new Random();
    List<int> list = new List<int>(length);
    Console.Write("Введіть мінімальне число масиву: ");
    int min = int.Parse(Console.ReadLine());
    Console.Write("Введіть максимальне число масиву: ");
    int max = int.Parse(Console.ReadLine());

    for (int i = 0; i < length; i++)
    {
        list.Add(random.Next(min, max));
    }

    return list;
}

static List<int> InputListSeparateLines()
{
    Console.Write("Введіть кількість елементів масиву: ");
    int length = int.Parse(Console.ReadLine());

    List<int> list = new List<int>(length);

    Console.WriteLine("Введіть елементи масиву по одному у кожному рядку:");
    for (int i = 0; i < length; i++)
    {
        Console.Write($"Елемент {i + 1}: ");
        list.Add(int.Parse(Console.ReadLine()));
    }

    return list;
}

static List<int> InputListSingleLine()
{
    Console.Write("Введіть елементи масиву, розділені пробілами або табуляціями: ");
    string[] input = Console.ReadLine().Split(new char[] { ' ', '\t' }, StringSplitOptions.RemoveEmptyEntries);

    List<int> list = new List<int>(input.Length);

    for (int i = 0; i < input.Length; i++)
    {
        list.Add(int.Parse(input[i]));
    }

    return list;
}

static void PrintList(List<int> list)
{
    Console.WriteLine(string.Join(", ", list));
}
static void DoBlock_2()
{
    Console.WriteLine("Введіть натуральне число n:");
    int n = Convert.ToInt32(Console.ReadLine());

    // Вимірювання пам'яті для пункту а)
    long memoryBefore = GC.GetTotalMemory(true);
    int[][] jaggedArray = CreateJaggedArray(n);
    long memoryAfter = GC.GetTotalMemory(true);
    long memoryUsedA = memoryAfter - memoryBefore;

    Console.WriteLine("Пункт а):");
    PrintJaggedArray2(jaggedArray);

    // Вимірювання пам'яті для пункту б)
    memoryBefore = GC.GetTotalMemory(true);
    var optimizedJaggedArray = CreateOptimizedJaggedArray(n);
    memoryAfter = GC.GetTotalMemory(true);
    long memoryUsedB = memoryAfter - memoryBefore;

    Console.WriteLine("Пункт б):");
    PrintOptimizedJaggedArray(n, optimizedJaggedArray);
```

```
    // Виведення використаної пам'яті
    Console.WriteLine("Використання пам'яті в пункті а): " + memoryUsedA + " байт");
    Console.WriteLine("Використання пам'яті в пункті б): " + memoryUsedB + " байт");
    Console.WriteLine();
}
// Функція для обчислення суми цифр числа
static int SumOfDigits(int number)
{
    int sum = 0;
    while (number > 0)
    {
        sum += number % 10;
        number /= 10;
    }
    return sum;
}

// Пункт а): Створення зубчастого масиву
static int[][] CreateJaggedArray(int n)
{
    int[][] jaggedArray = new int[n][];
    for (int i = 0; i < n; i++)
    {
        int sumDigits = SumOfDigits(i);
        if (sumDigits == 0)
        {
            jaggedArray[i] = new int[] { 0 };
            continue;
        }

        List<int> multiples = new List<int>();
        for (int j = 1; j <= n; j++)
        {
            if (j % sumDigits == 0)
            {
                multiples.Add(j);
            }
        }
        jaggedArray[i] = multiples.ToArray();
    }
    return jaggedArray;
}

// Пункт б): Створення зубчастого масиву з оптимізацією пам'яті
static int[][] CreateOptimizedJaggedArray(int n)
{
    // Створюємо зубчастий масив для зберігання послідовностей чисел, кратних кожній можливій сумі цифр
    // Максимальна сума цифр для числа до 999999 (9*6 = 54, але беремо з запасом, тому використовуємо 46)
    int[][] sumsArray = new int[46][];

    // Створюємо зубчастий масив, де кожен індекс буде вказувати на відповідну послідовність чисел
    int[][] optimizedJaggedArray = new int[n][];

    // Проходимося по всіх числах від 0 до n-1
    for (int i = 0; i < n; i++)
    {
        // Обчислюємо суму цифр поточного числа i
        int sumDigits = SumOfDigits(i);

        // Якщо сума цифр дорівнює 0 (наприклад, число 0), зберігаємо масив з єдиним елементом 0
        if (sumDigits == 0)
        {
            optimizedJaggedArray[i] = new int[] { 0 };
            continue; // Переходимо до наступного числа
        }

        // Якщо для даної суми цифр ще не створено послідовність чисел
        if (sumsArray[sumDigits] == null)
        {
            // Створюємо тимчасовий масив для зберігання чисел, кратних сумі цифр
            int[] multiples = new int[n];
            int count = 0; // Лічильник кількості кратних чисел

            // Проходимося по всіх числах від 1 до n
            for (int j = 1; j <= n; j++)
            {
                // Якщо число j кратне сумі цифр, додаємо його до тимчасового масиву
                if (j % sumDigits == 0)
                {
                    multiples[count++] = j;
                }
            }

            // Змінюємо розмір тимчасового масиву до фактичної кількості кратних чисел
            Array.Resize(ref multiples, count);

            // Зберігаємо цей масив в sumsArray для відповідної суми цифр
```

```
            sumsArray[sumDigits] = multiples;
        }

        // Встановлюємо посилання на відповідний масив з sumsArray у optimizedJaggedArray
        optimizedJaggedArray[i] = sumsArray[sumDigits];
    }

    // Повертаємо заповнений оптимізований зубчастий масив
    return optimizedJaggedArray;
}


// Виведення зубчастого масиву
static void PrintJaggedArray2(int[][] jaggedArray)
{
    for (int i = 0; i < jaggedArray.Length; i++)
    {
        Console.Write(i + ": ");
        foreach (var num in jaggedArray[i])
        {
            Console.Write(num + " ");
        }
        Console.WriteLine();
    }
}

// Виведення оптимізованого зубчастого масиву
static void PrintOptimizedJaggedArray(int n, int[][] optimizedJaggedArray)
{
    for (int i = 0; i < n; i++)
    {
        Console.Write(i + ": ");
        foreach (var num in optimizedJaggedArray[i])
        {
            Console.Write(num + " ");
        }
        Console.WriteLine();
    }
}

static void DoBlock_3()
{
    int[][] jaggedArray = InputJaggedArray();

    int maxRowIndex = FindRowWithMaxElement(jaggedArray);

    jaggedArray = AddRowAfterMaxRow(jaggedArray, maxRowIndex);

    Console.WriteLine("\nЗубчастий масив після додавання рядка:");
    PrintJaggedArray(jaggedArray);
    Console.WriteLine();
}

static int[][] ReadJaggedArrayManually()
{
    Console.WriteLine("Введіть розмірність зубчастого масиву (кількість рядків):");
    int rows = int.Parse(Console.ReadLine());

    int[][] jaggedArray = new int[rows][];

    for (int i = 0; i < rows; i++)
    {
        Console.WriteLine($"Введіть рядок {i + 1} (елементи через пробіл):");
        string[] elements = Console.ReadLine().Split(' ');
        jaggedArray[i] = Array.ConvertAll(elements, int.Parse);
    }

    return jaggedArray;
}

static int[][] GenerateRandomJaggedArray()
{
    Random random = new Random();

    Console.WriteLine("Введіть мінімальне значення:");
    int minValue = int.Parse(Console.ReadLine());

    Console.WriteLine("Введіть максимальне значення:");
    int maxValue = int.Parse(Console.ReadLine());

    Console.WriteLine("Введіть кількості елементів кожного рядка через пробіл:");
    string[] elementsCountsStr = Console.ReadLine().Split(' ');

    int rows = elementsCountsStr.Length;
    int[][] jaggedArray = new int[rows][];

    for (int i = 0; i < rows; i++)
    {
```

```
        int elementsCount = int.Parse(elementsCountsStr[i]);
        jaggedArray[i] = new int[elementsCount];

        for (int j = 0; j < elementsCount; j++)
        {
            jaggedArray[i][j] = random.Next(minValue, maxValue + 1);
        }
    }

    return jaggedArray;
}

static int FindRowWithMaxElement(int[][] jaggedArray)
{
    int maxRowIndex = 0;
    int maxElement = jaggedArray[0][0];

    for (int i = 0; i < jaggedArray.Length; i++)
    {
        for (int j = 0; j < jaggedArray[i].Length; j++)
        {
            if (jaggedArray[i][j] > maxElement)
            {
                maxElement = jaggedArray[i][j];
                maxRowIndex = i;
            }
        }
    }

    return maxRowIndex;
}

static int[][] AddRowAfterMaxRow(int[][] jaggedArray, int maxRowIndex)
{
    // Створюємо новий зубчастий масив, який має на один рядок більше ніж оригінальний масив jaggedArray.
    int[][] newArray = new int[jaggedArray.Length + 1][];

    // Копіювання елементів до maxRowIndex включно вручну
    for (int i = 0; i <= maxRowIndex; i++)
    {
        newArray[i] = jaggedArray[i];
    }

    // Вставити новий рядок після maxRowIndex
    newArray[maxRowIndex + 1] = new int[] { 0 };

    // Копіювання решти елементів вручну
    for (int i = maxRowIndex + 1; i < jaggedArray.Length; i++)
    {
        newArray[i + 1] = jaggedArray[i];
    }

    return newArray;
}

static void PrintJaggedArray(int[][] jaggedArray)
{
    for (int i = 0; i < jaggedArray.Length; i++)
    {
        for (int j = 0; j < jaggedArray[i].Length; j++)
        {
            Console.Write(jaggedArray[i][j] + " ");
        }
        Console.WriteLine();
    }
}
static int[][] InputJaggedArray()
{
    int[][] jaggedArray;

    Console.WriteLine("Оберіть спосіб введення зубчастого масиву:");
    Console.WriteLine("1. Вручну.");
    Console.WriteLine("2. Випадково введені значення.");

    int choice = int.Parse(Console.ReadLine());

    if (choice == 1)
        jaggedArray = ReadJaggedArrayManually();
    else
        jaggedArray = GenerateRandomJaggedArray();

    Console.WriteLine("\nВаш зубчастий масив:");
    PrintJaggedArray(jaggedArray);

    return jaggedArray;
}
static void DoBlock_7()
{
```

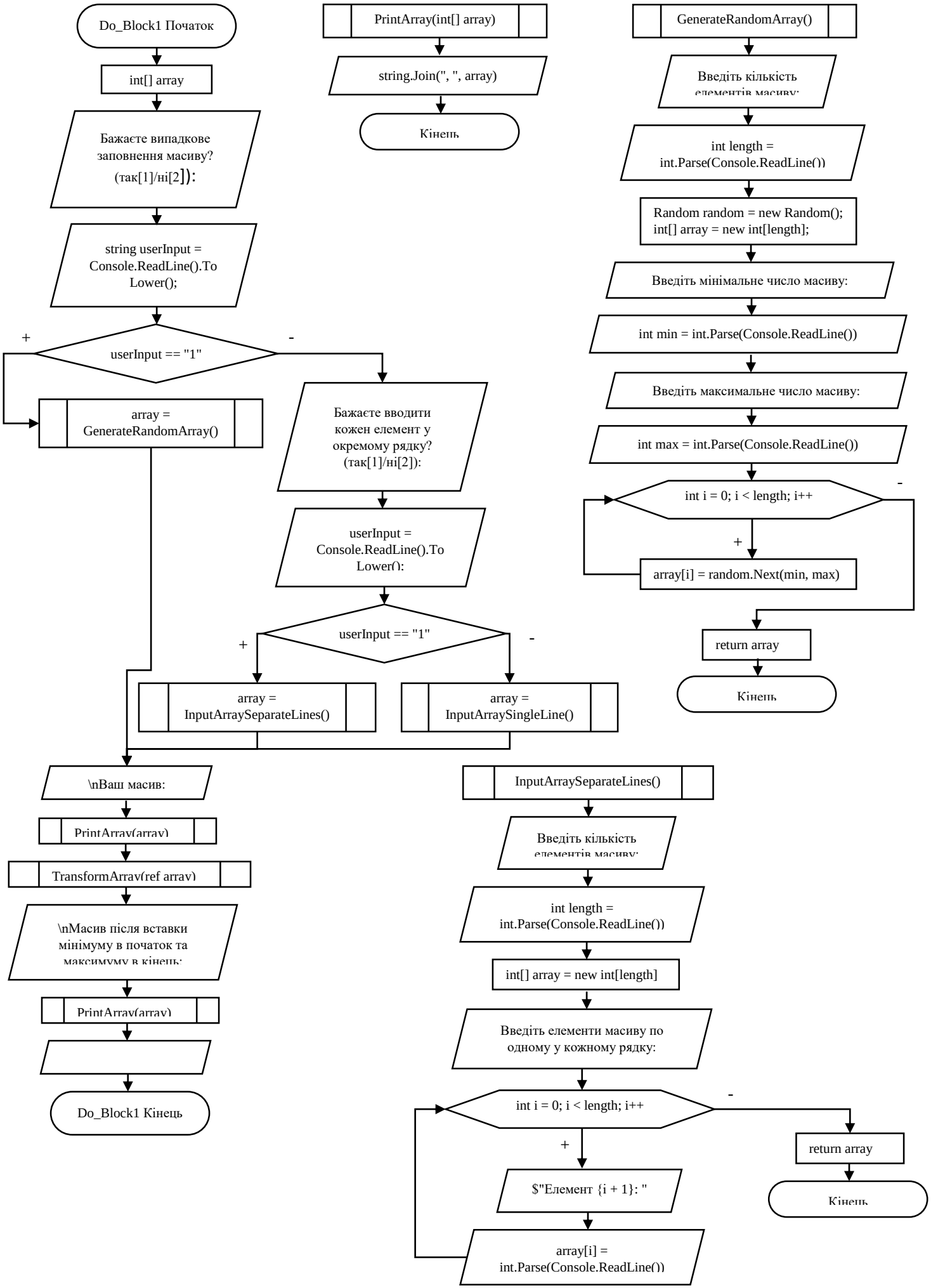
```
    List<List<int>> jaggedList = InputJaggedList();

    int maxRowIndex = FindRowWithMaxElement1(jaggedList);

    AddRowAfterMaxRow1(jaggedList, maxRowIndex);

    Console.WriteLine("\nЗубчастий список після додавання рядка:");
    PrintJaggedList(jaggedList);
    Console.WriteLine();
}

static List<List<int>> ReadJaggedListManually()
{
    Console.WriteLine("Введіть розмірність зубчастого списку (кількість рядків):");
    int rows = int.Parse(Console.ReadLine());

    List<List<int>> jaggedList = new List<List<int>>(rows);

    for (int i = 0; i < rows; i++)
    {
        Console.WriteLine($"Введіть рядок {i + 1} (елементи через пробіл):");
        string[] elements = Console.ReadLine().Split(' ');
        List<int> row = new List<int>(Array.ConvertAll(elements, int.Parse));
        jaggedList.Add(row);
    }

    return jaggedList;
}

static List<List<int>> GenerateRandomJaggedList()
{
    Random random = new Random();

    Console.WriteLine("Введіть мінімальне значення:");
    int minValue = int.Parse(Console.ReadLine());

    Console.WriteLine("Введіть максимальне значення:");
    int maxValue = int.Parse(Console.ReadLine());

    Console.WriteLine("Введіть кількості елементів кожного рядка через пробіл:");
    string[] elementsCountsStr = Console.ReadLine().Split(' ');

    int rows = elementsCountsStr.Length;
    List<List<int>> jaggedList = new List<List<int>>(rows);

    for (int i = 0; i < rows; i++)
    {
        int elementsCount = int.Parse(elementsCountsStr[i]);
        List<int> row = new List<int>(elementsCount);

        for (int j = 0; j < elementsCount; j++)
        {
            row.Add(random.Next(minValue, maxValue + 1));
        }

        jaggedList.Add(row);
    }

    return jaggedList;
}

static int FindRowWithMaxElement1(List<List<int>> jaggedList)
{
    int maxRowIndex = 0;
    int maxElement = jaggedList[0][0];

    for (int i = 0; i < jaggedList.Count; i++)
    {
        for (int j = 0; j < jaggedList[i].Count; j++)
        {
            if (jaggedList[i][j] > maxElement)
            {
                maxElement = jaggedList[i][j];
                maxRowIndex = i;
            }
        }
    }

    return maxRowIndex;
}

static void AddRowAfterMaxRow1(List<List<int>> jaggedList, int maxRowIndex)
{
    List<int> newRow = new List<int> { 0 };
    jaggedList.Insert(maxRowIndex + 1, newRow);
}

static void PrintJaggedList(List<List<int>> jaggedList)
```

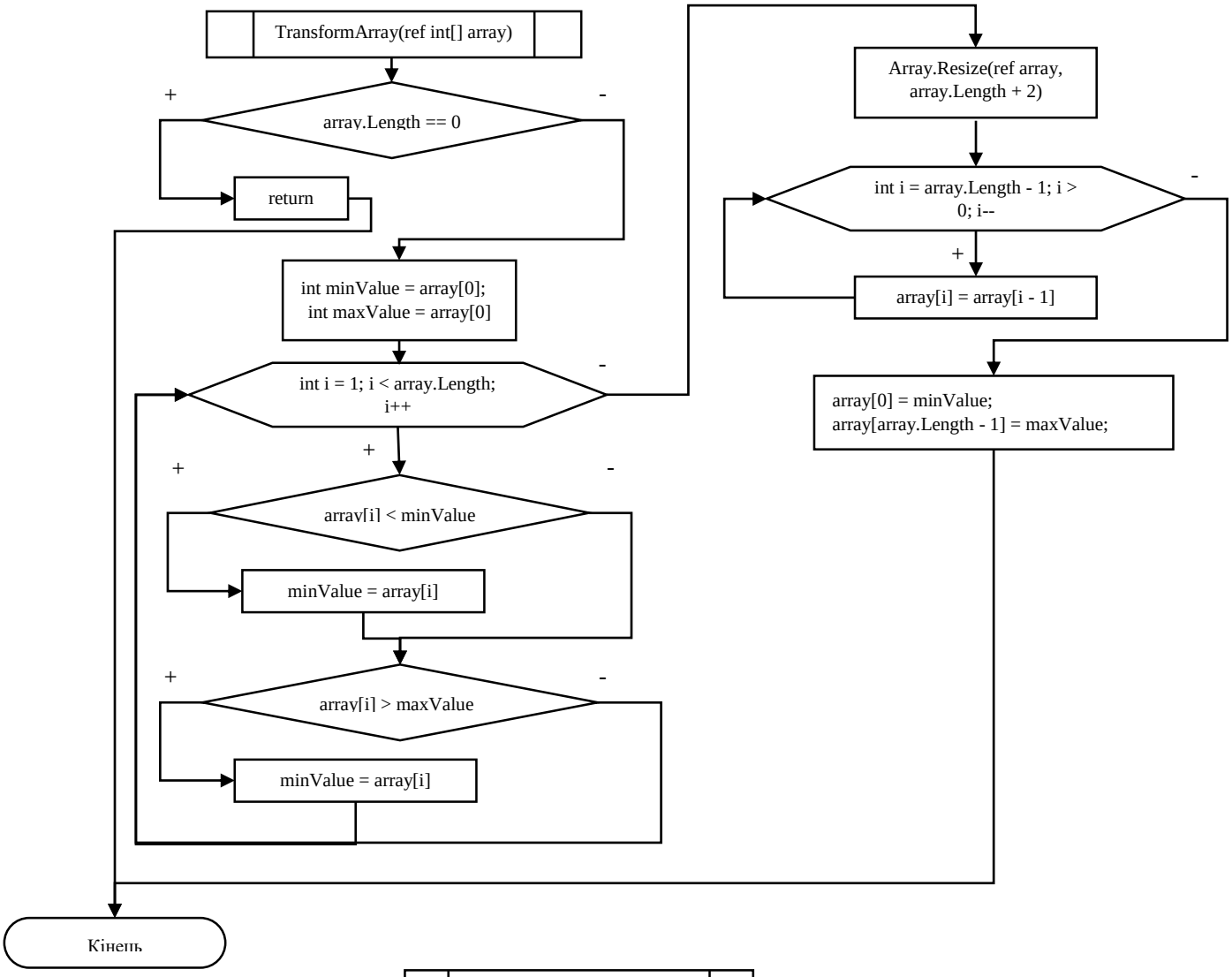
```
    {
        for (int i = 0; i < jaggedList.Count; i++)
        {
            for (int j = 0; j < jaggedList[i].Count; j++)
            {
                Console.Write(jaggedList[i][j] + " ");
            }
            Console.WriteLine();
        }
    }

    static List<List<int>> InputJaggedList()
    {
        List<List<int>> jaggedList;

        Console.WriteLine("Оберіть спосіб введення зубчастого списку:");
        Console.WriteLine("1. Вручну.");
        Console.WriteLine("2. Випадково введені значення.");

        int choice = int.Parse(Console.ReadLine());

        if (choice == 1)
            jaggedList = ReadJaggedListManually();
        else
            jaggedList = GenerateRandomJaggedList();

        Console.WriteLine("\nВаш зубчастий список:");
        PrintJaggedList(jaggedList);

        return jaggedList;
    }
    static void DoBlock_4()
    {
        int[][] jaggedArray = InputJaggedArray();

        // Переписати з кожного рядка зубчастого массива P у відповідні рядки зубчастого массива Q лише непарні
елементи.
        int[][] jaggedArrayQ = ExtractOddElements(jaggedArray);

        // Відсортувати (методом вибору) рядки отриманого зубчастого массива Q за зростанням.
        SortJaggedArrayRows(jaggedArrayQ);

        Console.WriteLine("\nЗубчастий масив Q з непарними елементами:");
        PrintJaggedArray(jaggedArrayQ);

        // Сформувати з рядків зубчастого массива Q одновимірний масив
        int[] oneDimensionalArray = FlattenJaggedArray(jaggedArrayQ);

        Console.WriteLine("\nОдновимірний масив:");
        Console.WriteLine(string.Join(" ", oneDimensionalArray));

        // Визначити ті його елементи, значення яких співпадають із власним індексом
        List<int> matchingElements = FindElementsMatchingIndices(oneDimensionalArray);

        Console.WriteLine("\nЕлементи, значення яких співпадають із власним індексом:");
        Console.WriteLine(string.Join(" ", matchingElements));
    }

    static int[][] ExtractOddElements(int[][] jaggedArray)
    {
        // Створюємо новий зубчастий масив результатів з тією ж кількістю рядків, що і вхідний масив.
        int[][] result = new int[jaggedArray.Length][];

        // Проходимо через кожен рядок вхідного зубчастого масиву.
        for (int i = 0; i < jaggedArray.Length; i++)
        {
            // Створюємо тимчасовий список для зберігання непарних елементів поточного рядка.
            List<int> oddElements = new List<int>();

            // Проходимо через кожен елемент поточного рядка.
            for (int j = 0; j < jaggedArray[i].Length; j++)
            {
                // Якщо елемент непарний (ділення на 2 дає залишок 1), додаємо його до списку непарних елементів.
                if (jaggedArray[i][j] % 2 != 0)
                {
                    oddElements.Add(jaggedArray[i][j]);
                }
            }

            // Перетворюємо список непарних елементів в масив і зберігаємо його в результаті для поточного рядка.
            result[i] = oddElements.ToArray();
        }

        // Повертаємо новий зубчастий масив, який містить тільки непарні елементи з кожного рядка вхідного масиву.
        return result;
    }

    static void SortJaggedArrayRows(int[][] jaggedArray)
```

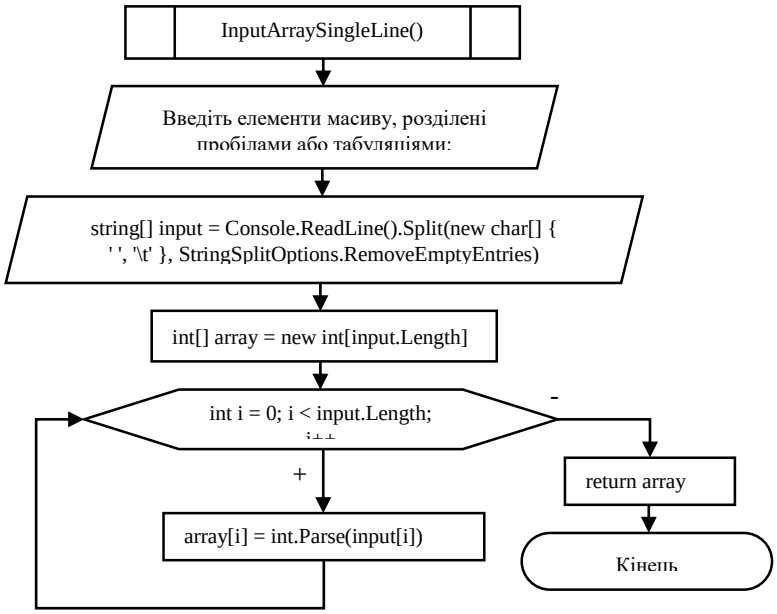
```
    {
        for (int i = 0; i < jaggedArray.Length; i++)
        {
            SelectionSort(jaggedArray[i]);
        }
    }

    static void SelectionSort(int[] array)
    {
        for (int i = 0; i < array.Length - 1; i++)
        {
            int minIndex = i;
            for (int j = i + 1; j < array.Length; j++)
            {
                if (array[j] < array[minIndex])
                {
                    minIndex = j;
                }
            }
            int temp = array[minIndex];
            array[minIndex] = array[i];
            array[i] = temp;
        }
    }

    static int[] FlattenJaggedArray(int[][] jaggedArray)
    {
        List<int> list = new List<int>();
        foreach (var array in jaggedArray)
        {
            list.AddRange(array);
        }
        return list.ToArray();
    }

    static List<int> FindElementsMatchingIndices(int[] array)
    {
        List<int> matchingElements = new List<int>();
        for (int i = 0; i < array.Length; i++)
        {
            if (array[i] == i)
            {
                matchingElements.Add(array[i]);
            }
        }
        return matchingElements;
    }

    static void Main(string[] args)
    {
        System.Threading.Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo("en-US");
        Console.OutputEncoding = UTF8Encoding.UTF8;
        Console.ForegroundColor = ConsoleColor.Black;
        Console.BackgroundColor = ConsoleColor.White;
        Console.Clear();
        int choice;
        do
        {
            Console.WriteLine("-Для виконання блоку 1 (11. Вставити в початок масиву мінімум з усіх значень масиву,
а в кінець — максимум введіть 1");
            Console.WriteLine("-Для виконання блоку 2 введіть 2");
            Console.WriteLine("-Для виконання блоку 3 (11. Додати рядок після рядка, що містить максимальний
елемент (якщо у різних місцях є кілька елементів з однаковим максимальним значенням, то брати перший з них) введіть
3");
            Console.WriteLine("-Для виконання блоку 4 (11. Переписати з кожного рядка матриці P у відповідні рядки
матриці Q лише непарні елементи.Відсортувати рядки отриманої матриці Q за зростанням. Сформувати з рядків матриці Q
одновимірний масив та визначити ті його елементи, значення яких співпадають із власним індексом) введіть 4");
            Console.WriteLine("-Для виконання блоку 1.1 (блок 1 бібліотечні методи) введіть 5");
            Console.WriteLine("-Для виконання блоку 1.2 (блок 1 List + методи List) введіть 6");
            Console.WriteLine("-Для виконання блоку 3.1 (блок 3 List) введіть 7");
            Console.WriteLine("-Для виходу з програми введіть 0");

            choice = int.Parse(Console.ReadLine());

            switch (choice)
            {
                case 1:
                    Console.WriteLine("Виконую блок 1");
                    DoBlock_1();
                    break;
                case 2:
                    Console.WriteLine("Виконую блок 2");
                    DoBlock_2();
                    break;
                case 3:
                    Console.WriteLine("Виконую блок 3");
                    DoBlock_3();
                    break;
```

```
                case 4:
                    Console.WriteLine("Виконую блок 4");
                    DoBlock_4();
                    break;
                case 5:
                    Console.WriteLine("Виконую блок 1.1");
                    DoBlock_5();
                    break;
                case 6:
                    Console.WriteLine("Виконую блок 1.2");
                    DoBlock_6();
                    break;
                case 7:
                    Console.WriteLine("Виконую блок 3.1");
                    DoBlock_7();
                    break;
                case 0:
                    Console.WriteLine("Зараз завершимо, тільки натисніть будь ласка ще раз Enter");
                    Console.ReadLine();
                    break;
                default:
                    Console.WriteLine("Команда ``{0}'' не розпізнана. Зробіь, будь ласка, вибір із 1, 2, 3, 4, 0.",
choice);
                    break;
            }
        } while (choice != 0);
    }
}
```

БЛОК – СХЕМА:


Do_Block1 Початок
int[] array
Бажаєте випадкове заповнення масиву? (так[1]/ні[2]):
string userInput = Console.ReadLine().ToLower();
userInput == "1"
+
-
array = GenerateRandomArray()
Бажаєте вводити кожен елемент у окремому рядку? (так[1]/ні[2]):
userInput = Console.ReadLine().ToLower();
userInput == "1"
+
-
array = InputArraySeparateLines()
array = InputArraySingleLine()
\nВаш масив:
PrintArray(array)
TransformArray(ref array)
\nМасив після вставки мінімуму в початок та максимуму в кінець:
PrintArray(array)
Do_Block1 Кінець
PrintArray(int[] array)
string.Join(", ", array)
Кінець
GenerateRandomArray()
Введіть кількість елементів масиву:
int length = int.Parse(Console.ReadLine())
Random random = new Random();
int[] array = new int[length];
Введіть мінімальне число масиву:
int min = int.Parse(Console.ReadLine())
Введіть максимальне число масиву:
int max = int.Parse(Console.ReadLine())
int i = 0; i < length; i++
+
-
array[i] = random.Next(min, max)
return array
Кінець
InputArraySeparateLines()
Введіть кількість елементів масиву:
int length = int.Parse(Console.ReadLine())
int[] array = new int[length]
Введіть елементи масиву по одному у кожному рядку:
int i = 0; i < length; i++
+
-
$"Елемент {i + 1}: "
array[i] = int.Parse(Console.ReadLine())
return array
Кінець


TransformArray(ref int[] array)
array.Length == 0
+
-
return
int minValue = array[0];
int maxValue = array[0]
int i = 1; i < array.Length; i++
array[i] < minValue
minValue = array[i]
array[i] > maxValue
minValue = array[i]
Array.Resize(ref array, array.Length + 2)
int i = array.Length - 1; i > 0; i--
array[i] = array[i - 1]
array[0] = minValue;
array[array.Length - 1] = maxValue;
Кінець


InputArraySingleLine()
Введіть елементи масиву, розділені пробілами або табуляціями:
string[] input = Console.ReadLine().Split(new char[] { ' ', '\t' }, StringSplitOptions.RemoveEmptyEntries)
int[] array = new int[input.Length]
int i = 0; i < input.Length;
array[i] = int.Parse(input[i])
return array
Кінець