Масиви и Функции в C#

Масиви: Съхранение на списъци от данни от един и същи тип.
Функции (Методи): Именовани блокове, които изпълняват специфична задача и се преизползват.
Дефиниране на масив: **int**[] **numbers** = **new** int[5];
Структура на метод: **static** [тип] ***Име***([параметри]) { ... }
Комбинираме ги за да се обработ групи данни по организиран начин (сортиране, търсене, изчисления).
Проста функция, която приема масив като аргумент:
Тази функция изчислява сумата на всички елементи в масив.

```
using System;
class Program
{
    static void Main()
    {
        int[] myNumbers = { 10, 20, 30, 40, 50 };
        // Извикване на функцията и подаване на масива
        int result = CalculateSum(myNumbers);
        Console.WriteLine($"Сумата на елементите е: {result}");
    }
    // Дефиниране на функцията
    static int CalculateSum(int[] array)
    {
        int sum = 0;
        foreach (int number in array)
        {
            sum += number;
        }
        return sum;
    }
}
```

Функция, която променя масив (***Reference Type***):
Масивите се предават по референция – ако ги променим във функцията, те се променят и в **Main**

```
static void DoubleValues(int[] array)
{
    for (int i = 0; i < array.Length; i++)
    {
        array[i] *= 2; // Умножаваме всеки елемент по 2
    }
}
```

Ключови моменти за запомняне:
Синтаксис: Когато подаваме масив, в параметрите на функцията пишем типа с квадратни скоби: **int**[] ***arrayName***.
Дължина: Винаги използваме **array.Length** вътре във функциите, за да не излезем извън границите на масива.
Връщане на масив: Една функция може и да връща нов масив, например: **static int**[] **GenerateArray**(***int size***).

Пример, за масив с цикъл (**for** и **foreach**) вътре във функции с две функции:
една за принтиране на масива и една за намиране на средно аритметично.

```
using System;
class Program
{
    static void Main()
    {
        // 1. Създаваме масив с оценки
        double[] grades = { 5.50, 4.00, 6.00, 3.50, 5.00 };
        Console.WriteLine("Списък с оценки:");
        // Извикваме функцията за принтиране
        PrintGrades(grades);
        // 2. Изчисляваме средния успех чрез функция
        double average = GetAverage(grades);
        Console.WriteLine($"\nСреден успех: {average:F2}");
    }
```

```
    // Функция, която използва FOREACH цикъл (за четене)
    static void PrintGrades(double[] array)
    {
        foreach (double grade in array)
        {
            Console.Write(grade + " ");
        }
        Console.WriteLine();
    }
    // Функция, която използва FOR цикъл (за изчисления)
    static double GetAverage(double[] array)
    {
        double sum = 0;
        for (int i = 0; i < array.Length; i++)
        {
            sum += array[i]; // добавяме всеки елемент към сумата
        }
        return sum / array.Length; // връщаме средната стойност
    }
}
```

Какво се случва тук?

PrintGrades: Използваме **foreach**, преминем през всички елементи, за да ги покажем само (без да ги променяме).

GetAverage: Използваме стандартен **for** цикъл. Чрез **array.Length** програмата върти точно толкова пъти, колкото са елементите в масива.

Параметърът: И в двата случая подаваме масива просто като име (**grades**), а функцията го приема като тип **double[]**.

Пример, в който Потребителят определя колко числа ще въведе. Използваме цикъл, за да запълним масива. Използваме функция, за да обработим готовия масив.

Задача променете броя на масива!

```
using System;
class Program
{
    static void Main()
    {
        // 1. Питаме за размера на масива
        Console.Write("Колко числа ще въведете? ");
        int count = int.Parse(Console.ReadLine());
        // 2. Създаваме масива с посочения размер
        int[] numbers = new int[count];
        // 3. Цикъл за въвеждане на числата от потребителя
        for (int i = 0; i < numbers.Length; i++)
        {
            Console.Write($"Въведете число {i + 1}: ");
            numbers[i] = int.Parse(Console.ReadLine());
        }
        // 4. Извикваме функция, за да намерим най-малкото число
        int minNumber = FindMin(numbers);
        Console.WriteLine("\nГотово!");
        Console.WriteLine($"Най-малкото число в масива е: {minNumber}");
    }
    // Функция, която приема масива и намира минималната стойност
    static int FindMin(int[] array)
    {
        int min = array[0]; // Приемаме, че първото число е най-малко
        for (int i = 1; i < array.Length; i++)
        {
            if (array[i] < min)
            {
                min = array[i]; // Ако намерим по-малко, го записваме
            }
        }
        return min;     }    }
```

Как работи този код:
new int[count]: Резервира място в паметта за точно толкова числа, колкото е написал потребителят.
numbers[i] = ...: Използваме индекса **i**, за да поставим всяко въведено число на правилната му „кутийка" в масива.
Логиката в **FindMin:** Обхождаме масива и сравняваме всеки елемент с текущия най-малък.

СОРТИРАНЕ

1. **Bubble Sort (Метод на мехурчето)**

```
static void BubbleSort(int[] array)
{
    int n = array.Length;
    for (int i = 0; i < n - 1; i++)
    {
        // Последната част на масива вече е сортирана, затова ползваме - i
        for (int j = 0; j < n - i - 1; j++)
        {
            if (array[j] > array[j + 1])
            {
                // Класическа размяна (Swap) чрез временна променлива
                int temp = array[j];
                array[j] = array[j + 1];
                array[j + 1] = temp;
            }
        }
    }
}
```

2. Selection Sort (Пряка селекция)

```
static void SelectionSort(int[] array)
{
    int n = array.Length;
    for (int i = 0; i < n - 1; i++)
    {
        int minIndex = i; // Предполагаме, че текущият е най-малък
        for (int j = i + 1; j < n; j++)
        {
            if (array[j] < array[minIndex])
            {
                minIndex = j; // Намерили сме по-малък елемент
            }
        }
        // Разменяме намерения най-малък елемент с текущия
        int temp = array[minIndex];
        array[minIndex] = array[i];
        array[i] = temp;
    }
}
```

Основна разлика:
Bubble Sort: Прави много размени (сменя местата на съседни числа непрекъснато).
Selection Sort: Прави само една размяна на всяко пълно преминаване през масива, което го прави малко по-ефективен при запис в паметта.

Пример **A**
въвеждане на масив,
меню за избор на алгоритъм
и използване на **функции**

```
using System;
class Program
{
    static void Main()
    {
        // 1. Въвеждане на данни от потребителя
        Console.Write("Въведете брой елементи в масива: ");
        int n = int.Parse(Console.ReadLine());
        int[] numbers = new int[n];
        for (int i = 0; i < n; i++)
        {
            Console.Write($"Елемент [{i}]: ");
            numbers[i] = int.Parse(Console.ReadLine());
        }
        // 2. Меню за избор
        Console.WriteLine("\nИзберете метод за сортиране:");
        Console.WriteLine("1 - Bubble Sort (Метод на мехурчето)");
        Console.WriteLine("2 - Selection Sort (Пряка селекция)");
        Console.Write("Вашият избор: ");
        string choice = Console.ReadLine();
        // 3. Изпълнение на избрания алгоритъм
        if (choice == "1")
        {
            BubbleSort(numbers);
            Console.WriteLine("\nМасивът е сортиран чрез Bubble Sort:");
        }
        else if (choice == "2")
        {
            SelectionSort(numbers);
            Console.WriteLine("\nМасивът е сортиран чрез Selection Sort:");
        }
        else
```

```
        {
            Console.WriteLine("\nНевалиден избор. Масивът няма да бъде сортиран.");
        }
        // 4. Извеждане на резултата чрез функция
        PrintArray(numbers);
    }
    // --- ФУНКЦИИ ---
    static void BubbleSort(int[] array)
    {
        int n = array.Length;
        for (int i = 0; i < n - 1; i++)
        {
            for (int j = 0; j < n - i - 1; j++)
            {
                if (array[j] > array[j + 1])
                {
                    // Размяна (Swap)
                    int temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
            }
        }
    }
    static void SelectionSort(int[] array)
    {
        int n = array.Length;
        for (int i = 0; i < n - 1; i++)
        {
            int minIndex = i;
            for (int j = i + 1; j < n; j++)
            {
```

```
                if (array[j] < array[minIndex])
                {
                    minIndex = j;
                }
            }
            // Размяна
            int temp = array[minIndex];
            array[minIndex] = array[i];
            array[i] = temp;
        }
    }
    static void PrintArray(int[] array)
    {
        foreach (int item in array)
        {
            Console.Write(item + " ");
        }
        Console.WriteLine();
    }
}
```

Пример **Б**
въвеждане на масив,
меню за избор
двата алгоритъма за сортиране, реализирани като отделни функции.
и трета опция в менюто – използването на професионалния вграден метод **Array.Sort()**:

```
using System;
class Program
{
    static void Main()
    {
        // 1. Въвеждане на масива
        Console.Write("Въведете брой елементи: ");
        int n = int.Parse(Console.ReadLine());
        int[] numbers = new int[n];
        for (int i = 0; i < n; i++)
        {
            Console.Write($"Елемент [{i}]: ");
            numbers[i] = int.Parse(Console.ReadLine());
        }
        // 2. Меню с вградения метод
        Console.WriteLine("\nИзберете метод за сортиране:");
        Console.WriteLine("1 - Bubble Sort");
        Console.WriteLine("2 - Selection Sort");
        Console.WriteLine("3 - Вграден метод (Array.Sort)");
        Console.Write("Вашият избор: ");
        string choice = Console.ReadLine();
        // 3. Логика за избор
        switch (choice)
        {
            case "1":
                BubbleSort(numbers);
                Console.WriteLine("\nСортирано чрез Bubble Sort:");
                break;
            case "2":
                SelectionSort(numbers);
                Console.WriteLine("\nСортирано чрез Selection Sort:");
                break;
            case "3":
                Array.Sort(numbers); // Вграденият метод на C#
                Console.WriteLine("\nСортирано чрез Array.Sort():");
                break;
```

```
            default:
                Console.WriteLine("\nНевалиден избор.");
                break;
        }
        // 4. Извеждане на резултата
        PrintArray(numbers);
    }
    // --- Помощни функции ---
    static void BubbleSort(int[] array)
    {
        for (int i = 0; i < array.Length - 1; i++)
            for (int j = 0; j < array.Length - i - 1; j++)
                if (array[j] > array[j + 1])
                {
                    int temp = array[j];
                    array[j] = array[j + 1];
                    array[j + 1] = temp;
                }
    }
    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 void PrintArray(int[] array)
    {
        Console.WriteLine(string.Join(", ", array));
    }
```