

**Министерство науки и высшего образования Российской Федерации**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**  
**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**  
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**  
**ITMO University**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5**

**По дисциплине** Объектно-ориентированное программирование

**Тема работы** Создание и использование массивов

**Обучающийся** Буров Глеб Максимович

**Факультет** факультет инфокоммуникационных технологий

**Группа** K3223

**Направление подготовки** 11.03.02 Инфокоммуникационные технологии и системы связи

**Образовательная программа** Программирование в инфокоммуникационных системах

**Обучающийся**

\_\_\_\_\_  
(дата)

\_\_\_\_\_  
(подпись)

Буров Г.М.  
(Ф.И.О.)

**Руководитель**

\_\_\_\_\_  
(дата)

\_\_\_\_\_  
(подпись)

Иванов С.Е.  
(Ф.И.О.)

# СОДЕРЖАНИЕ

Стр.

<b>ВВЕДЕНИЕ .....</b>	<b>3</b>
<b>1   Ход работы .....</b>	<b>4</b>
1.1   Упражнение 1 .....	4
1.2   Упражнение 2 .....	5
1.3   Упражнение 3 .....	7
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>12</b>

## **ВВЕДЕНИЕ**

Целью данной лабораторной работы является изучение массивов и приобретению навыков работы с ними.

## 1 Ход работы

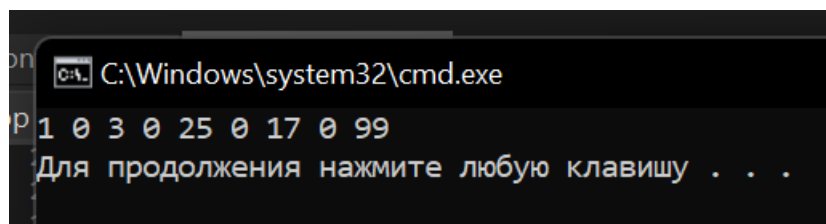
### 1.1 Упражнение 1

В первой части этого упражнения необходимо реализовать массив для хранения данных (`myArray`), после чего вывести все нечётные элементы массива, а четные заменить на 0 (рис. 1.1).

```
int[] myArray = { 100, 1, 32, 3, 14, 25, 6, 17, 8, 99 };  
for (int i = 1; i <= myArray.Length; i++)  
{  
    if (myArray[i] % 2 == 0)  
        myArray[i] = 0;  
    Console.Write(myArray[i] + " ");  
}  
Console.WriteLine();
```

Рисунок 1.1 — Код упражнения 1

Пример работы метода показан на рис. 1.2.

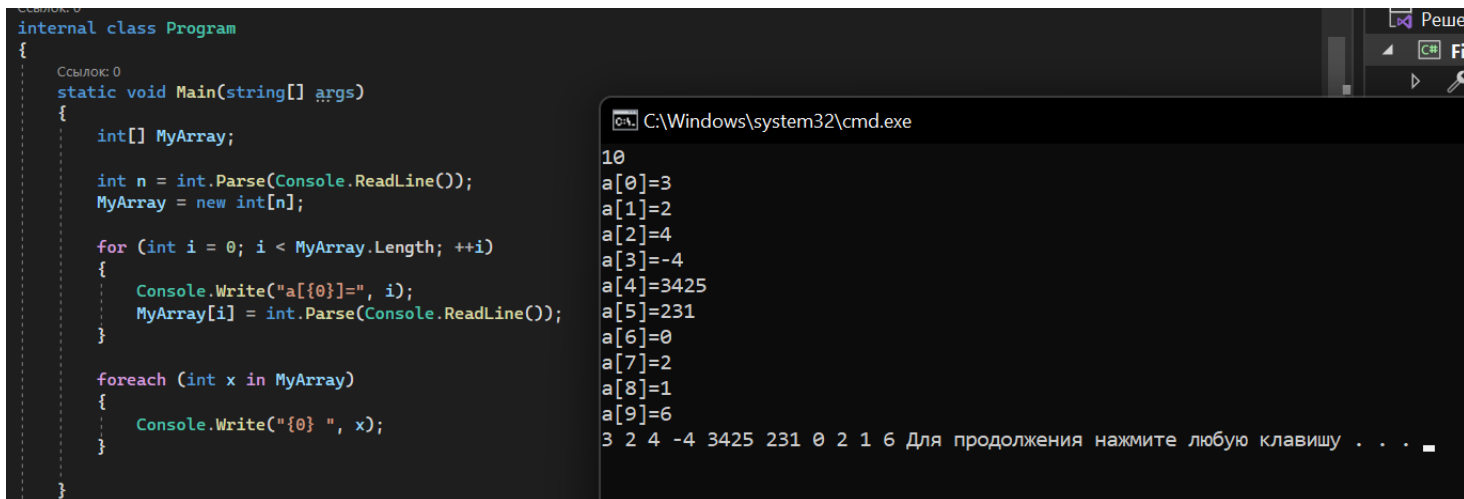


```
C:\Windows\system32\cmd.exe  
1 0 3 0 25 0 17 0 99  
Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.2 — Пример работы программы из упражнения 1

Дальше нужно было заменить массив с явной инициализацией на массив, размер которого вводит с клавиатуры пользователь. После ввода размера пользователь также должен проинициализировать каждый элемент массива (осуществляется с помощью цикла `for`).

Код программы и пример работы показаны на рис. 1.3.



The image shows a C# program in Visual Studio and its execution in the Windows Command Prompt. The program defines an array `MyArray` and reads 10 integers from the console. The console output shows the array elements being read and then printed.

```
internal class Program
{
    static void Main(string[] args)
    {
        int[] MyArray;

        int n = int.Parse(Console.ReadLine());
        MyArray = new int[n];

        for (int i = 0; i < MyArray.Length; ++i)
        {
            Console.Write("a[{0}]=", i);
            MyArray[i] = int.Parse(Console.ReadLine());
        }

        foreach (int x in MyArray)
        {
            Console.Write("{0} ", x);
        }
    }
}
```

```
C:\Windows\system32\cmd.exe
10
a[0]=3
a[1]=2
a[2]=4
a[3]=-4
a[4]=3425
a[5]=231
a[6]=0
a[7]=2
a[8]=1
a[9]=6
3 2 4 -4 3425 231 0 2 1 6 Для продолжения нажмите любую клавишу . . .
```

Рисунок 1.3 — Инициализация массива пользователем

## 1.2 Упражнение 2

Во втором упражнении нужно было написать программу, в которой массивы будут использоваться для перемножения матриц. Программа должна считывать с консоли 4 целых числа и сохранять их в матрице размером 2x2, затем будут считаны еще 4 целых числа и сохранены в еще одной матрице размером 2x2. Далее эти матрицы будут перемножаться, а результат будет сохранен в третьей матрице таких же размеров. Результат умножения должен быть выведен на экран консоли.

Для удобства инициализация матрицы значениями с консоли, умножение матриц, а также операция вывода матрицы на экран были вынесены в отдельные методы — `Input`, `Multiply` и `Output` (рис. 1.4).

```

private static void Input(int[,] a)
{
    for (int r = 0; r < a.GetLength(0); r++)
    {
        for (int c = 0; c < a.GetLength(1); c++)
        {
            Console.Write("Enter value for [{0},{1}] : ", r, c);
            string s = System.Console.ReadLine();
            a[r, c] = int.Parse(s);
        }
    }
    Console.WriteLine();
}

Ссылка: 1
private static int[,] Multiply(int[,] a, int[,] b)
{
    int[,] result = new int[2, 2];

    for (int r = 0; r < result.GetLength(0); r++)
    {
        for (int c = 0; c < result.GetLength(1); c++)
        {
            result[r, c] += a[r, 0] * b[0, c] + a[r, 1] * b[1, c];
        }
    }
    return result;
}

Ссылка: 1
private static void Output(int[,] result)
{
    for (int r = 0; r < result.GetLength(0); r++)
    {
        for (int c = 0; c < result.GetLength(1); c++)
        {
            Console.Write("{0} ", result[r, c]);
        }
        Console.WriteLine();
    }
}

```

Рисунок 1.4 — Код упражнения 2

Пример работы метода показан на рис. 1.5.

<pre> Ссылка: 0 internal class MatrixMultiply {     Ссылка: 0     static void Main(string[] args)     {         int[,] a = new int[2, 2];         Input(a);          int[,] b = new int[2, 2];         Input(b);          int[,] result = Multiply(a, b);         Output(result);     } }  Ссылка: 2 </pre>	<pre> C:\Windows\system32\cmd.exe Enter value for [0,0] : 3 Enter value for [0,1] : 2 Enter value for [1,0] : 4 Enter value for [1,1] : 4  Enter value for [0,0] : 6 Enter value for [0,1] : 0 Enter value for [1,0] : 3 Enter value for [1,1] : 2  24 4 36 8 </pre>
---	--

Рисунок 1.5 — Пример работы программы из упражнения 2

### 1.3 Упражнение 3

В рамках последнего упражнения требовалось самостоятельно реализовать заполнение массива с клавиатуры, добавив в программу методы для обработки данных массива: определение суммы всех элементов массива, определение среднего значения элементов массива, расчет суммы отрицательных или положительных элементов, расчет суммы элементов с четными или нечетными номерами, нахождение индекса максимального или минимального элемента массива, расчет произведения элементов, находящихся между максимальным и минимальным.

Для реализации заполнения массива с клавиатуры был создан метод `Fill`, имеющий такой же функционал, как и метод `Input` в предыдущем задании.

Для нахождения суммы элементов массива был реализован метод `Sum`, имеющий единственный аргумент — массив элементов типа `double`. В методе с помощью цикла `foreach` к переменной `sum` добавляются элементы массива.

Для нахождения суммы отрицательных или положительных элементов массива метод `Sum` был перегружен — добавился второй аргумент `positive`, имеющий значение типа `bool`. Если на вход поступает значение `true`, суммируются только положительные элементы массива, если `false` — отрицательные (рис. 1.6).

```

private static double Sum(double[] arr)
{
    double sum = 0;
    foreach (double number in arr)
    {
        sum += number;
    }
    return sum;
}

Ссылка: 2
private static double Sum(double[] arr, bool positive)
{
    double sum = 0;
    if (positive)
    {
        foreach (double element in arr)
        {
            if (element > 0)
            {
                sum += element;
            }
        }
    }
    else
    {
        foreach (double element in arr)
        {
            if (element < 0)
            {
                sum += element;
            }
        }
    }
    return sum;
}

```

Рисунок 1.6 — Метод Sum и перегруженный метод Sum

Расчет суммы элементов с четными или нечетными номерами производится с помощью метода NumberSum, принимающий в качестве аргументов наш массив и переменную even типа bool. Перебор элементов массива происходит с помощью цикла for, начальное значение переменной i определяется значением even — если true, значит мы ищем сумму элементов с четными номерами (под номером я имею в виду порядковый номер элемента, а не его индекс), соответственно начать мы должны со второго элемента ( $i = 1$ ), в противном случае мы идем по нечетным номерам, начиная с первого ( $i = 0$ ). Шаг цикла равен двум.



Для определения среднего значения элементов массива был определен метод Average. В нем вызывается описанный выше метод Sum, и результат делится на длину массива (рис. 1.7).

```
private static double NumberSum(double[] arr, bool even)
{
    double sum = 0;
    int i;
    if (even)
    {
        i = 1;
    }
    else
    {
        i = 0;
    }

    for (; i < arr.Length; i += 2)
    {
        sum += arr[i];
    }

    return sum;
}

Ссылка: 1
private static double Average(double[] arr)
{
    double average;
    average = Sum(arr) / arr.Length;
    return average;
}
```

Рисунок 1.7 — Методы NumberSum и Average

Для нахождения минимального и максимального элемента массива использовались методы Min и Max соответственно (рис. 1.8).

```

private static int Min(double[] arr)
{
    double min = double.MaxValue;
    int index = -1;
    for (int i = 0; i < arr.Length; i++)
    {
        if (arr[i] < min)
        {
            min = arr[i];
            index = i;
        }
    }

    return index;
}

Ссылка: 1
private static int Max(double[] arr)
{
    double max = double.MinValue;
    int index = -1;
    for (int i = 0; i < arr.Length; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
            index = i;
        }
    }

    return index;
}

```

Рисунок 1.8 — Методы Min и Max

Чтобы рассчитать произведение элементов массива, находящихся между максимальным и минимальным элементами, был создан метод MultyBetweenMinMax, в котором изначально находятся индексы минимального и максимального элементов (с помощью вышеописанных методов), после чего условным оператором проверяется, соседние ли эти элементы — если да, то произведение элементов между ними равно 0 (элементов между min и max нет), в противном случае с помощью цикла for проходимся по элементам между и перемножаем (рис. 1.9).

```

private static double MultyBetweenMinMax(double[] arr)
{
    int index1 = Min(arr);
    int index2 = Max(arr);
    int temp;
    double ans = 1;

    if (index2 < index1)
    {
        temp = index2;
        index2 = index1;
        index1 = temp;
    }

    if (index2 - index1 == 1)
        ans = 0;

    for (int i = index1 + 1; i < index2; i++)
    {
        ans *= arr[i];
    }

    return ans;
}

```

Рисунок 1.9 — Метод MultyBetweenMinMax

Пример работы программы показан на рис. 1.10.

```

Console.WriteLine("Введите размер массива: ");
int n = int.Parse(Console.ReadLine());

double[] arr = new double[n];
Fill(arr, n);

```

C:\Windows\system32\cmd.exe

```

Введите размер массива:
5
arr[0] = 3
arr[1] = 10
arr[2] = 4
arr[3] = 2
arr[4] = -3
} Сумма элементов массива: 16
Среднее арифметическое элементов массива: 3,2
Сумма положительных элементов массива: 19
Сумма отрицательных элементов массива: -3
Сумма элементов массива с четными номерами: 12
Сумма элементов массива с нечетными номерами: 4
Индекс минимального элемента массива: 4
Индекс максимального элемента массива: 1
Произведение элементов массива, находящихся между min и max: 8
Для продолжения нажмите любую клавишу . . .

```

Рисунок 1.10 — Пример работы программы из упражнения 3

## **ЗАКЛЮЧЕНИЕ**

В ходе этой лабораторной работы я узнал о работе с обычными массивами, а также массивами с несколькими измерениями, изучил, как устроен рефакторинг в среде разработки Visual Studio.NET. Все задания были выполнены.