# Лабораторная работа № 3 массивы и Строки

**Цель лабораторной работы**

Закрепление теоретических знаний при работе с массивами и строками.

**Постановка задачи**

Массив - набор элементов одного и того же типа, объединенных общим именем. Массивы в С# можно использовать по аналогии с тем, как они используются в других языках программирования. Однако С#-массивы имеют существенные отличия: они относятся к ссылочным типам данных, более того - реализованы как объекты. Фактически имя массива является ссылкой на область кучи (динамической памяти), в которой последовательно размещается набор элементов определенного типа. Выделение памяти под элементы происходит на этапе инициализации массива. А за освобождением памяти следит система сборки мусора - неиспользуемые массивы автоматически утилизируются данной системой. Рассмотрим различные типы массивов.

**Одномерные массивы**

Одномерный массив - это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в С# начинается с нуля, то есть, если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Одномерный массив в С# реализуется как объект, поэтому его создание представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная на массив, затем выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве. Базовый тип определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяется размер массива. В общем случае процесс объявления переменной типа массив, и выделение необходимого объема памяти может быть разделено. Кроме того, на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

|  |  |
| --- | --- |
| Форма записи | Пояснения |
| базовый\_тип [] имя\_\_массива;  Например:  int [] a; | Описана ссылка на одномерный массив, которая в дальнейшем может быть использована: 1. для адресации на уже существующий массив; 2. передачи массива в метод в качестве параметра 3. отсроченного выделения памяти под элементы массива. |
| базовый\_тип [] имя\_\_массива = new базовый\_тип [размер];  Например:  int []a=new int [10]; | Объявлен одномерный массив заданного типа и выделена память под одномерный массив указанной размерности. Адрес данной области памяти записан в ссылочную переменную. Элементы массива равны нулю. Замечание. Надо отметить, что в C# элементам массива присваиваются начальные значения по умолчанию в зависимости от базового типа. Для арифметических типов - нули, для ссылочных типов - null, для символов - пробел. |
| базовый\_тип [] имя\_\_массива= {список инициализации};  Например:  int []a={0, 1, 2, 3}; | Выделена память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в ссылочную переменную. Значение элементов массива соответствует списку инициализации. |

Обращения к элементам массива происходи с помощью индекса, для этого нужно указать имя массива и в квадратных скобках его номер. Например, a[0], b[10], c[i]. Так как массив представляет собой набор элементов, объединенных общим именем, то обработка массива обычно производится в цикле. Рассмотрим несколько простых примеров работы с одномерными массивами.

**Многомерные массивы**

Многомерные массивы имеют более одного измерения. Чаще всего используются двумерные массивы, которые представляют собой таблицы. Каждый элемент массива имеет два индекса, первый определяет номер строки, второй - номер столбца, на пересечении которых находится элемент. Нумерация строк и столбцов начинается с нуля.

Объявить двумерный массив можно одним из предложенных способов:

тип [,] имя\_\_массива;

тип [,] имя\_\_массива = new тип [размер1, размер2];

тип [,] имя\_\_массива={

{элементы 1-ой строки},

… ,

{элементы n-ой строки}};

тип [,] имя\_\_массива= new тип [,]

{

{элементы 1-ой строки},

… ,

{элементы n-ой строки}

};

Например:

int [,] a;

int [,] a= new int [3, 4];

int [,] a=

{

{0, 1, 2},

{3, 4, 5}

};

int [,] a= new int [,]

{

{0, 1, 2},

{3, 4, 5}

};

**Замечания.**

1. Как и в случае с одномерными массивами, последние два описания являются избыточными.

2. При работе с многомерными массивами можно использовать приемы, которые мы рассмотрели для одномерных массивов.

3. При обращении к свойству Length для двумерного массива мы получим общее количество элементов в массиве. Чтобы получить количество строк нужно обратиться к методу GetLength с параметром 0. Чтобы получить количество столбцов – к методу GetLength с параметром 1.

**Оператор foreach и его использование при работе с массивами**

Оператор foreach применяется для перебора элементов в специальным образом организованной группе данных, в том числе и в массиве. Удобство этого вида цикла заключается в том, что нам не требуется определять количество элементов в группе и выполнять перебор по индексу - мы просто указываем на необходимость перебрать все элементы группы.

Синтаксис оператора: foreach ( in ) где имя определяет локальную по отношению к циклу переменную, которая будет по очереди принимать все значения из указанной группы, а тип соответствует базовому типу элементов группы. Ограничением оператора foreach является то, что с его помощью можно только просматривать значения элементов в группе данных, но нельзя их изменять.

string[] fruits = { "яблоко", "банан", "апельсин", "виноград" };

// Перебор всех элементов массива

foreach (string fruit in fruits)

{

Console.WriteLine(fruit);

}

**Методы сортировки данных C#**

- Сортировка пузырьком;

- Шейкерная сортировка;

- Сортировка вставками;

- Гномья сортировка;

- Сортировка простым выбором;

- Поразрядная сортировка;

- Сортировка Шелла;

- Пирамидальная сортировка;

- Быстрая сортировка.

**Сортировка пузырьком**

Реализация на языке C#:

using System;

class Program

{

    static void Main()

    {

        int[] array = { 64, 34, 25, 12, 22, 11, 90 };

        Console.WriteLine("Исходный массив:");

        PrintArray(array);

        BubbleSort(array);

        Console.WriteLine("Отсортированный массив:");

        PrintArray(array);

    }

    static void BubbleSort(int[] arr)

    {

        int n = arr.Length;

        for (int i = 0; i < n - 1; i++)

        {

            // Последние i элементов уже отсортированы

            for (int j = 0; j < n - i - 1; j++)

            {

                // Сравниваем соседние элементы

                if (arr[j] > arr[j + 1])

                {

                    // Меняем их местами, если они в непра-вильном порядке

                    int temp = arr[j];

                    arr[j] = arr[j + 1];

                    arr[j + 1] = temp;

                }

            }

        }

    }

    static void PrintArray(int[] arr)

    {

        foreach (int value in arr)

        {

            Console.Write(value + " ");

        }

        Console.WriteLine();

    }

}

- простейший алгоритм сортировки. Эффективен для небольшого объёма данных, а при сортировке большого объема данных могут быть большие потери времени.

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N+1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде, отсюда и название алгоритма).

**Шейкерная сортировка** (сортировка перемешиванием)

- разновидность пузырьковой сортировки.

Перестановка элементов в шейкерной сортировке выполняется аналогично той же в пузырьковой сортировке, т. е. два соседних элемента, при необходимости, меняются местами. Суть модификации заключается в реализации двунаправленности: алгоритм перемещается, ни как в пузырьковой сортировке – строго снизу вверх (слева направо), а сначала снизу вверх, потом сверху вниз.

Реализация на языке C#:

using System;

class Program

{

    //метод обмена элементов

    static void Swap(ref int e1, ref int e2)

    {

        var temp = e1;

        e1 = e2;

        e2 = temp;

    }

    //сортировка перемешиванием

    static int[] ShakerSort(int[] array)

    {

        for (var i = 0; i < array.Length / 2; i++)

        {

            var swapFlag = false;

            //проход слева направо

            for (var j = i; j < array.Length - i - 1; j++)

            {

                if (array[j] > array[j + 1])

                {

                    Swap(ref array[j], ref array[j + 1]);

                    swapFlag = true;

                }

            }

            //проход справа налево

            for (var j = array.Length - 2 - i; j > i; j--)

            {

                if (array[j - 1] > array[j])

                {

                    Swap(ref array[j - 1],

ref array[j]);

                    swapFlag = true;

                }

            }

            //если обменов не было выходим

            if (!swapFlag)

            {

                break;

            }

        }

        return array;

    }

    static void Main(string[] args)

    {

        Console.WriteLine("Шейкерная сортировка");

        Console.Write("Введите элементы массива: ");

        var parts = Console.ReadLine().Split(

new[] { " ", ",", ";" },

StringSplitOptions.RemoveEmptyEntries);

        var array = new int[parts.Length];

        for (int i = 0; i < parts.Length; i++)

        {

            array[i] = Convert.ToInt32(parts[i]);

        }

        Console.WriteLine("Отсортированный массив: {0}", string.Join(", ", ShakerSort(array)));

        Console.ReadLine();

    }

}

**Сортировка вставками** (сортировка простыми включениями)

- алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

На вход алгоритма подаётся некая последовательность. На каждом шаге алгоритма выбирается один из элементов, входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан.

**Гномья сортировка**

Реализация на языке C#:

using System;

class Program

{

    //метод обмена элементов

    static void Swap(ref int e1, ref int e2)

    {

        var temp = e1;

        e1 = e2;

        e2 = temp;

    }

    //сортировка вставками

    static int[] InsertionSort(int[] array)

    {

        for (var i = 1; i < array.Length; i++)

        {

            var key = array[i];

            var j = i;

            while ((j > 1) && (array[j - 1] > key))

            {

                Swap(ref array[j - 1], ref array[j]);

                j--;

            }

            array[j] = key;

        }

        return array;

    }

    static void Main(string[] args)

    {

        Console.WriteLine("Сортировка вставками");

        Console.Write("Введите элементы массива: ");

        var parts = Console.ReadLine().Split(

new[] { " ", ",", ";" },

StringSplitOptions.RemoveEmptyEntries);

        var array = new int[parts.Length];

        for (int i = 0; i < parts.Length; i++)

        {

            array[i] = Convert.ToInt32(parts[i]);

        }

        Console.WriteLine("Упорядоченный массив: {0}", string.Join(", ", InsertionSort(array)));

        Console.ReadLine();

    }

}

- алгоритм сортировки, похожий на сортировку вставками, но в отличие от последней перед вставкой на нужное место происходит серия обменов, как в сортировке пузырьком

Алгоритм находит первое место, где два соседних элемента стоят в неправильном порядке и меняет их местами. Он пользуется тем фактом, что обмен может породить новую пару, стоящую в неправильном порядке, только до или после переставленных элементов.

**Сортировка выбором**

Шаги алгоритма:

1) находим номер минимального значения в текущем списке

2) производим обмен этого значения со значением первой неотсортированной позиции (обмен не нужен, если минимальный элемент уже находится на данной позиции)

3) теперь сортируем хвост списка, исключив из рассмотрения уже отсортированные элементы

**Поразрядная сортировка**

- алгоритм сортировки за линейное время. Алгоритм сортирует числа по разрядам.

Сортировка Шелла

Идея сортировки методом Шелла состоит в том, чтобы сортировать элементы отстоящие друг от друга на некотором расстоянии step. Затем сортировка повторяется при меньших значениях step, и в конце процесс сортировки Шелла завершается при step = 1 (а именно обычной сортировкой вставками).

**Пирамидальная сортировка**

Сортировка пирамидой использует бинарное сортирующее дерево. Сортирующее дерево — это такое дерево, у которого выполнены условия:

1) Каждый лист имеет глубину либо d, либо d-1; d — максимальная глубина дерева.

2) Значение в любой вершине не меньше (другой вариант — не больше) значения её потомков.

**Быстрая сортировка**

- Один из самых быстрых известных универсальных алгоритмов сортировки массивов.

Пошаговое описание алгоритма

1. Из массива выбирается элемент a[i]. Как правило, в качестве этого элемента берется центральный элемент массива.

2. Остальные элементы распределяются таким образом, чтобы слева от a[i] оказались все элементы, меньшие или равные a[i]. Элементы, большие или равные a[i], помещаются справа.

3. Производится проверка количества элементов в левой и правой частях массива. Если какая-либо часть (или обе части) содержит более двух элементов, то для этой части (или частей) запускается та же процедура сортировки с помощью рекурсивного вызова.

На небольшом объеме данных наиболее хорошие результаты показывают простейшие алгоритмы сортировки - вставками, пузырьком. Однако отличие во времени от остальных методов не очень существенно. На больших же объемах данных - сортировка пузырьком, шейкерная, гномья и, отчасти, сортировки вставками и выбором показывают существенное отличие во времени.

**Строки**

В языке C# строковые значения представляет тип string, а вся функциональность работы с данным типом сосредоточена в классе **System.String**. Собственно, string является псевдонимом для класса System.String. Объекты этого класса представляют текст как последовательность символов Unicode. Максимальный размер объекта String может составлять в памяти 2 ГБ, или около 1 миллиарда символов.

**Создание строк**

Создавать сроки можно, как используя переменную типа string и присваивая ей значение, так и применяя один из конструкторов класса String:

string s1 = "hello";

string s2 = null;

string s3 = new String('a', 6); // результатом будет строка "aaaaaa"

string s4 = new String(new char[]{'w', 'o', 'r', 'l', 'd'});

Конструктор String имеет различное число версий. Так, вызов конструктора new String('a', 6) создаст строку "aaaaaa". И так как строка представляет ссылочный тип, то может хранить значение null.

**Строка как набор символов**

Так как строка хранит коллекцию символов, в ней определен индексатор для доступа к этим символам.

Применяя индексатор, мы можем обратиться к строке как к массиву символов и получить по индексу любой из ее символов:

string s1 = "hello";

char ch1 = s1[1]; // символ 'e'

Console.WriteLine(ch1);

Console.WriteLine(s1.Length);

Используя свойство Length, как и в обычном массиве, можно получить длину строки.

**Основные методы строк**

Основная функциональность класса String раскрывается через его методы, среди которых можно выделить следующие:

* **Compare**: сравнивает две строки с учетом текущей культуры (локали) пользователя
* **CompareOrdinal**: сравнивает две строки без учета локали
* **Contains**: определяет, содержится ли подстрока в строке
* **Concat**: соединяет строки
* **CopyTo**: копирует часть строки или всю строку в другую строку
* **EndsWith**: определяет, совпадает ли конец строки с подстрокой
* **Format**: форматирует строку
* **IndexOf**: находит индекс первого вхождения символа или подстроки в строке
* **Insert**: вставляет в строку подстроку
* **Join**: соединяет элементы массива строк
* **LastIndexOf**: находит индекс последнего вхождения символа или подстроки в строке
* **Replace**: замещает в строке символ или подстроку другим символом или подстрокой
* **Split**: разделяет одну строку на массив строк
* **Substring**: извлекает из строки подстроку, начиная с указанной позиции
* **ToLower**: переводит все символы строки в нижний регистр
* **ToUpper**: переводит все символы строки в верхний регистр
* **Trim**: удаляет начальные и конечные пробелы из строки

Рассмотрим некоторые примеры работы со строками на C#.

Конкатенация строк:

string firstName = "Алексей";

string lastName = "Иванов";

// Конкатенация строк

string fullName = firstName + " " + lastName;

Console.WriteLine("Полное имя: " + fullName);

Извлечение подстроки:

string text = "Программирование на C#";

// Извлечение подстроки

string substring = text.Substring(0, 11); // "Программирова"

Console.WriteLine("Подстрока: " + substring);

Изменение регистра:

string text = "Привет, мир!";

// Изменение регистра

Console.WriteLine("В верхнем регистре: " + text.ToUpper());

Console.WriteLine("В нижнем регистре: " + text.ToLower());

Поиск и замена:

string text = "Привет, мир!";

// Поиск подстроки

int index = text.IndexOf("мир");

Console.WriteLine("Индекс 'мир': " + index);

// Замена подстроки

string newText = text.Replace("мир", "C#");

Console.WriteLine("После замены: " + newText);

Разделение строки:

string csv = "яблоко,банан,вишня";

// Разделение строки

string[] fruits = csv.Split(',');

Console.WriteLine("Фрукты:");

foreach (string fruit in fruits)

{

Console.WriteLine(fruit);

}

Форматирование строк:

string name = "Алексей";

int age = 30;

// Форматирование строки

string formattedString = string.Format("Меня зовут {0}, и мне {1} лет.", name, age);

Console.WriteLine(formattedString);

Использование интерполяции строк:

string name = "Алексей";

int age = 30;

// Интерполяция строк

string interpolatedString = $"Меня зовут {name}, и мне {age} лет.";

Console.WriteLine(interpolatedString);

Объединение строк с помощью Join:

string[] words = { "Программирование", "на", "C#", "это", "ин-тересно!" };

// Объединение строк с разделителем

string joinedString = string.Join(" ", words);

Console.WriteLine("Объединенные строки: " + joinedString);

Удаление пробелов:

string textWithSpaces = " Привет, мир! ";

// Удаление пробелов в начале и в конце строки

string trimmedText = textWithSpaces.Trim();

Console.WriteLine("После удаления пробелов: '" + trimmedText + "'");

Проверка на пустую строку:

string emptyString = "";

// Проверка, является ли строка пустой или состоит только из пробелов

bool isEmpty = string.IsNullOrWhiteSpace(emptyString);

Console.WriteLine("Строка пустая: " + isEmpty);

Сравнение строк:

string str1 = "Привет";

string str2 = "привет";

// Сравнение строк с учетом регистра

bool areEqual = str1.Equals(str2);

Console.WriteLine("Строки равны (с учетом регистра): " + areEqual);

// Сравнение строк без учета регистра

bool areEqualIgnoreCase = str1.Equals(str2, StringComparison.OrdinalIgnoreCase);

Console.WriteLine("Строки равны (без учета регистра): " + areEqualIgnoreCase);

Проверка наличия подстроки:

string text = "Программирование на C#";

// Проверка наличия подстроки

bool containsCSharp = text.Contains("C#");

Console.WriteLine("Содержит 'C#': " + containsCSharp);

Замена символов:

string text = "Привет, мир!";

// Замена символов

string replacedText = text.Replace("и", "и́");

Console.WriteLine("После замены: " + replacedText);

Получение длины строки:

string text = "Привет, мир!";

// Получение длины строки

int length = text.Length;

Console.WriteLine("Длина строки: " + length);

Преобразование строки в массив символов:

string text = "Привет";

// Преобразование строки в массив символов

char[] characters = text.ToCharArray();

Console.WriteLine("Символы в строке:");

foreach (char c in characters)

{

Console.WriteLine(c);

}

**Задание на лабораторную работу**

1. Разработать консольное приложение для выполнения заданий.
2. Реализовать меню для выбора задания (рисунок 1), обеспечивающее возможность многократного выполнения различных заданий.
3. После завершения выполнения выбранного задания пользователь должен возвращаться в меню для повторного выбора.
4. Завершение работы программы должно осуществляться при явном выборе соответствующего пункта меню (например, 'Выход').
5. Выполнить задания согласно варианту.
6. Большие наборы данных генерировать случайным образом.
7. **Не использовать** встроенные функции сортировки массива!
8. Подготовить входные данные для проверки работы приложения.

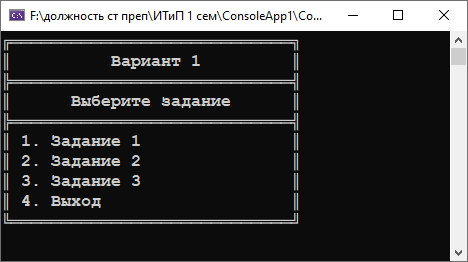


Рисунок 1 – Пример меню выбора задания

**Варианты заданий на лабораторную работу**

**Задание 1.**

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- сумму отрицательных элементов массива;

- произведение элементов массива, расположенных между максимальным и минимальным элементами.

Упорядочить элементы массива по возрастанию.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- сумму положительных элементов массива;

- произведение элементов массива, расположенных между максимальным по модулю и минимальным по модулю элементами.

Упорядочить элементы массива по убыванию.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- произведение элементов массива с четными номерами;

- сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Преобразовать массив таким образом, чтобы сначала располагались все положительные элементы, а потом – все отрицательные (элементы, равные нулю, считать положительными).

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- произведение элементов массива с нечетными номерами;

- сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Сжать массив, удалив из него все элементы, модуль которых не превышает единицу. Освободившиеся в конце массива элементы заполнить нулями.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- максимальный элемент массива;

- сумму элементов массива, расположенных до последнего положительного элемента.

Сжать массив, удалив из него все элементы, модуль которых находится в интервале [a, b]. Освободившиеся в конце массива элементы заполнить нулями.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- минимальный элемент массива;

- сумму элементов массива, расположенных между первым и последним положительными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом – все остальные.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- номер максимального элемента массива;

- произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине – элементы, стоявшие в четных позициях.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- номер минимального элемента массива;

- сумму элементов массива, расположенных между первым и вторым отрицательными элементами.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает единицу, а потом – все остальные.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- максимальный по модулю элемент массива;

- сумму элементов массива, расположенных между первым и вторым положительными элементами.

Преобразовать массив таким образом, чтобы элементы, равные нулю, располагались после всех остальных.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- минимальный по модулю элемент массива;

- сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- номер минимального по модулю элемента массива;

- сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Сжать массив, удалив из него все элементы, величина которых находится в интервале [a, b]. Освободившиеся в конце массива элементы заполнить нулями.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- номер максимального по модулю элемента массива;

- сумму элементов массива, расположенных после первого положительного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале [a, b], а потом – все остальные.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество элементов массива, лежащих в диапазоне от A до B;

- сумму элементов массива, расположенных после максимального элемента.

Упорядочить элементы массива по убыванию модулей.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество элементов массива, равных нулю;

- сумму элементов массива, расположенных после минимального элемента.

Упорядочить элементы массива по возрастанию модулей.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество элементов массива, больших C;

- произведение элементов массива, расположенных после максимального по модулю элемента.

Преобразовать массив таким образом, чтобы сначала располагались все отрицательные элементы, а потом – все положительные (элементы, равные нулю, считать положительными).

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество отрицательных элементов массива;

- сумму модулей элементов массива, расположенных после минимального по модулю элемента.

Заменить все отрицательные элементы массива их квадратами и упорядочить элементы массива по возрастанию.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество положительных элементов массива;

- сумму элементов массива, расположенных после последнего элемента, равного нулю.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает единицу, а потом – все остальные.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- количество элементов массива, меньших C;

- сумму целых частей элементов массива, расположенных после последнего отрицательного элемента.

Преобразовать массив таким образом, чтобы сначала располагались все элементы, отличающиеся от максимального не более чем на 20%, а потом – все остальные.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- произведение отрицательных элементов массива;

- сумму положительных элементов массива, расположенных до максимального элемента.

Изменить порядок следования элементов в массиве на обратный.

1. В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- произведение положительных элементов массива;

- сумму элементов массива, расположенных до минимального элемента.

Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

**Задание 2.**

1. Дана целочисленная прямоугольная матрица. Определить:

- количество строк, не содержащих ни одного нулевого элемента;

- максимальное из чисел, встречающихся в заданной матрице более одного раза.

1. Дана целочисленная прямоугольная матрица. Определить количество столбцов, не содержащих не одного нулевого элемента. Характеристикой строки целочисленной матрицы назовем сумму ее положительных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с ростом характеристик.
2. Дана целочисленная прямоугольная матрица. Определить:

- количество столбцов, содержащих хотя бы один нулевой элемент;

- номер строки, в которой находится самая длинная серия одинаковых элементов.

1. Дана целочисленная квадратная матрица. Определить:

- произведение элементов в тех строках, которые не содержать отрицательных элементов;

- максимум среди сумм элементов диагоналей. параллельных главной диагонали матрицы.

1. Дана целочисленная квадратная матрица. Определить:

- сумму элементов в тех столбцах, которые не содержать отрицательных элементов;

- минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

1. Дана целочисленная прямоугольная матрица. Определить:

- сумму элементов в тех строках, которые содержат хотя бы один отрицательных элемент;

- номера строк и столбцов всех седловых точек матрицы.

Матрица А имеет седловую точку Aij, если Aij является минимальным элементом в i-й строке и максимальным – в j-м столбце.

1. Для заданной матрицы размером 8x8 найти такие k, при которых k-я строка матрицы совпадает с k-м столбцом. Найти сумму элементов в тех строках, которые содержать хотя бы один отрицательный элемент.
2. Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствии с ростом характеристик. Найти сумму элементов в тех столбцах, которые содержать хотя бы один отрицательный элемент.
3. Соседями элемента Aij в матрице назовем элементы Akl, где i – 1 ≤ k ≤ i + 1, j – 1 ≤ l ≤ j + 1, (k,l) ≠ (i,j). Операция сглаживания матрицы дает новую матрицу того же размера, каждый элемент которой получается как среднее арифметическое имеющихся соседей соответствующего элемента исходной матрицы. Построить результат сглаживания заданной вещественной матрицы размером 10x10.
4. Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей (определение соседних элементов см. в варианте 9). Подсчитать количество локальных минимумов заданной матрицы размером 10x10. Найти сумму модулей элементов, расположенных выше главной диагонали.
5. Коэффициенты системы линейных уравнений заданы в виде прямоугольной матрицы. С помощью допустимых преобразований привести систему к треугольному виду. Найти количество строк, среднее арифметическое элементов которых меньше заданной величины.
6. Уплотнить заданную матрицу, удаляя из нее строки и столбцы, заполненные нулями. Найти номер первой из строк, содержащих хотя бы один положительный элемент.
7. Осуществить циклический сдвиг элементов прямоугольной матрицы на n элементов вправо или вниз (в зависимости от введенного режима). n может быть больше количества элементов в строке или столбце.
8. Осуществить циклический сдвиг элементов квадратной матрицы размером MxN вправо на k элементов таким образом: элементы первой строки сдвигаются в последний столбец сверху вниз, из него – в последнюю строку справа налево, из нее – в первый столбец снизу вверх, из него – в первую строку; для остальных элементов – аналогично.
9. Дана целочисленная прямоугольная матрица. Определить номер первого из столбцов, содержащих хотя бы один нулевой элемент. Характеристикой строки целочисленной матрицы назовем сумму ее отрицательных четных элементов. Переставляя строки заданной матрицы, расположить их в соответствии с убыванием характеристик.
10. Упорядочить строки целочисленной прямоугольной матрицы по возрастанию количества одинаковых элементов в каждой строке. Найти номер первого из столбцов, не содержащих ни одного отрицательного элемента.
11. Путем перестановки элементов квадратной вещественной матрицы добиться того, чтобы ее максимальный элемент находился в левой верхнем углу, следующий по величине – в позиции (2, 2), следующих по величине – в позиции (3, 3) и т.д., заполнив таким образом всю главную диагональ. Найти номер первой из строк, не содержащих ни одного положительного элемента.
12. Дана целочисленная прямоугольная матрица. Определить:

- количество строк, содержащих хотя бы один нулевой элемент;

- номер столбца, в котором находится самая длинная серия одинаковых элементов.

1. Дана целочисленная квадратная матрица. Определить:

- сумму элементов в тех строках, которые не содержат отрицательных элементов;

- минимум среди сумм элементов диагоналей, параллельных главной диагонали матрицы.

1. Дана целочисленная прямоугольная матрица. Определить:

- количество отрицательных элементов в тех строках, которые содержать хотя бы один нулевой элемент;

- номера строк и столбцов всех седловых точек матрицы.

Матрица А имеет седловую точку Aij, если Aij является минимальным элементом в i-й строке и максимальным – в j-м столбце.

**Задание 3.**

1. Расстояние между двумя словами равной длины – это количество позиций, в которых различаются эти слова. В заданном предложении найти пару слов с максимальным расстоянием.
2. Отредактировать заданное предложение, удаляя из него все слова с нечетными номерами и переворачивая слова с четными номерами.
3. Задан текст. Переписать его таким образом, чтобы предложения шли в обратном порядке, при этом внутри предложения слова шли в обратном порядке. Каждое слово в предложении должно начинаться с заглавной буквы. Предложения в новом тексте должны заканчиваться теми же знаками препинания, что и в исходном тексте.
4. Задан текст. Переписать его таким образом, чтобы сначала шли все гласные буквы, затем знаки препинания, а затем согласные буквы в том порядке в каком они встречаются в тексте.
5. Задан текст. Удалить из него все знаки препинания и переставить в нем пробелы таким образом, чтобы длина всех слов, кроме быть может последнего, была одинаковой.
6. Задан текст. Поменять в нем местами две гласные, которые встречаются чаще остальных двумя согласными, которые встречаются чаще остальных.
7. Дана строка, содержащая текст, вывести на экран те слова которые читаются одинаково справа налево и слева направо (т.е. являются палиндромом).
8. Упорядочить слова заданного текста по алфавиту.
9. Задан текст. Удалить из него все знаки препинания. Дописать в его конец минимальное количество символов так, чтобы полученный текст был палиндромом (пробелы не учитывать).
10. Ввести строку, в которую могут входить только цифры и символы {', ', '; ' , '\*', '@'}. Вывести на экран сколько раз встречается последовательность символов ',@,'
11. Ввести строку, в которой могут встречаться только буквы и цифры. Вывести на экран такие группы, в которых цифра 1 встречается не менее 3 раз. Под группой считать такую последовательность цифр, которая заключена между буквами.
12. Из введенного текста вывести на экран только те слова, которые симметричны (RADAR, ANNA)
13. Из введенного текста вывести на экран только те слова, в которых буквы упорядочены в порядке, обратном алфавитному (ROD, SON).
14. Сколько чисел между n и m (n<m) состоит из нечётных цифр и сколько из различных цифр?
15. Задан текст, содержащий только буквы. Переписать его таким образом, чтобы сначала шли все гласные буквы, а затем согласные буквы в том порядке в каком они встречаются алфавите.
16. Отредактировать заданное предложение, удаляя из него все слова с, входящей в них произвольной буквой (буква задается пользователем), и переворачивая в обратном порядке слова без вхождения этой буквы.
17. Строка содержит произвольный текст. Проверить совпадают ли в ней самые короткие слова (короткое слово – это слово с наименьшим количеством символов).
18. Задан текст. Удалить из него все знаки препинания и пробелы. Дописать в его конец столько раз слово 'ой!' сколько символов было удалено.
19. Задан текст. Дописать к слову длинной, заданной пользователем, слово, которое вводит пользователь. Все остальные -удалить.
20. Из введенного текста вывести на экран только те слова, в которых буквы упорядочены в алфавитном порядке (dor, nos).
21. Ввести строку, в которой могут встречаться только буквы и цифры. Вывести на экран такие группы, в которых последовательность букв 'ок' встречается не менее 2 раз. Под группой считать такую последовательность букв, которая заключена между цифрами.
22. Будем считать расстоянием между двумя одинаковыми словами – количество символов между ними. В заданном предложении найти расстояние между всеми соседними одинаковыми словами.
23. Даны две строки А и В. Составьте программу, выводящую на экран совпадающие подстроки длиной 2.
24. Строка содержит произвольный текст. Проверить совпадают ли в ней самые короткие слова (короткое слово – это слово с наименьшим количеством символов).
25. Задан текст. Удалить из него самое длинное и самое короткое слово. Во всех остальных словах вхождение заданной пользователем буквы увеличить вдвое.
26. Дан текст. Найти самое короткое слово, заменить его словом «вот!». В остальных словах текста заменить первые символы на «вот!», оставшиеся символы – не изменять.
27. Будем считать расстоянием между двумя словами – количество символов между этими словами. В заданном предложении найти расстояние между первым и последним вхождениями слова, введенного пользователем.

**Пример разработки приложения**

**Задание 1.**

В одномерном массиве, состоящем из *n* вещественных элементов, вычислить:

- сумму положительных элементов массива;

- произведение элементов массива, расположенных до минимального элемента.

Упорядочить по убыванию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

**Шаг 1. Создание проекта в Visual Studio 2022**

1. Запустите **Visual Studio 2022**
2. Нажмите **"Создать новый проект"**
3. Выберите шаблон **"Консольное приложение"** (C#)
4. Укажите:
   * Имя проекта: ArrayOperations
   * Расположение: выберите папку для сохранения
5. Нажмите **"Создать"**

**Шаг 2. Написание кода программы**

В файле Program.cs оформим код в виде методов.

**1. Ввод размера массива**

int InputArraySize()

{

Console.Write("Введите размер массива n: ");

return int.Parse(Console.ReadLine());

}

**Что делает:**

* Запрашивает у пользователя размер массива
* Преобразует введенную строку в целое число
* Возвращает размер массива

**2. Ввод элементов массива**

Запишем метод для ввода элементов массива:

void InputArrayElements(double[] array)

{

Console.WriteLine("Введите элементы массива:");

for (int i = 0; i < array.Length; i++)

{

array[i] = double.Parse(Console.ReadLine());

}

}

**Что делает:**

* Последовательно запрашивает каждый элемент массива
* Заполняет массив введенными значениями

**3. Вычисление суммы положительных элементов**

Запишем метод для вычисления суммы положительных элементов:

double CalculatePositiveSum(double[] array)

{

double sum = 0;

foreach (var item in array)

{

if (item > 0)

{

sum += item;

}

}

return sum;

}

**Что делает:**

* Перебирает все элементы массива
* Суммирует только положительные элементы
* Возвращает полученную сумму

**4. Вычисление произведения элементов до минимального**

Запишем метод для вычисления произведения элементов до минимального:

double CalculateProductBeforeMin(double[] array)

{

double product = 1;

int minIndex = 0;

double minValue = array[0];

for (int i = 1; i < array.Length; i++)

{

if (array[i] < minValue)

{

minValue = array[i];

minIndex = i;

}

}

for (int i = 0; i < minIndex; i++)

{

product \*= array[i];

}

return product;

}

**Что делает:**

* Находит минимальный элемент и его индекс
* Вычисляет произведение всех элементов до минимального
* Возвращает полученное произведение

**5. Сортировка элементов на четных и нечетных позициях**

Запишем метод для упорядочивания элементов на четных и нечетных местах:

void SortArrayByIndex(double[] array)

{

// Определяем количество четных и нечетных индексов

int evenCount = (array.Length + 1) / 2;

int oddCount = array.Length / 2;

double[] evenIndexedElements = new double[evenCount];

double[] oddIndexedElements = new double[oddCount];

// Заполнение массивов четных и нечетных индексов

for (int i = 0; i < array.Length; i++)

{

if (i % 2 == 0)

{

evenIndexedElements[i / 2] = array[i];

}

else

{

oddIndexedElements[i / 2] = array[i];

}

}

// Сортировка по убыванию четных индексов

SortDescending(evenIndexedElements);

// Сортировка по убыванию нечетных индексов

SortDescending(oddIndexedElements);

// Заполнение исходного массива отсортированными элемента-ми

for (int i = 0; i < evenCount; i++)

{

array[i \* 2] = evenIndexedElements[i];

}

for (int i = 0; i < oddCount; i++)

{

array[i \* 2 + 1] = oddIndexedElements[i];

}

}

**6. Вспомогательный метод сортировки по убыванию**

Запишем метод для сортировки по убыванию:

void SortDescending(double[] arr)

{

for (int i = 0; i < arr.Length - 1; i++)

{

for (int j = i + 1; j < arr.Length; j++)

{

if (arr[i] < arr[j])

{

// Обмен значениями

double temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

}

}

Запишем метод для вывода массива:

void PrintArray(double[] array)

{

Console.WriteLine(string.Join(", ", array));

}

Применим разработанные методы и решим задачу:

int n = InputArraySize();

// Инициализация массива

double[] array = new double[n];

// Ввод элементов массива

InputArrayElements(array);

// 1. Сумма положительных элементов

double positiveSum = CalculatePositiveSum(array);

Console.WriteLine($"Сумма положительных элементов: {posi - tiveSum}");

// 2. Произведение элементов до минимального

double productBeforeMin = CalculateProductBeforeMin(array);

Console.WriteLine($"Произведение элементов до минимального: {productBeforeMin}");

// 3. Упорядочивание элементов на четных и нечетных местах

SortArrayByIndex(array);

// Вывод отсортированного массива

Console.WriteLine("Отсортированный массив:");

PrintArray(array);

**Шаг 3. Запуск и тестирование программы**

1. Нажмите **F5** для запуска программы
2. Введите:
   * Размер массива (например, 5)
   * Элементы массива (например: 2, -1, 4, 0, 3)
3. Проверьте вывод:
   * Сумму положительных элементов (2 + 4 + 3 = 9)
   * Произведение до минимального элемента (2 \* -1 = -2)
   * Отсортированный массив (четные и нечетные позиции отдельно)

**Задание 2.**

Дана целочисленная прямоугольная матрица. Определить:

- количество положительных элементов в тех строках, которые содержат хотя бы два нулевых элемент;

- номера строк и столбцов всех седловых точек матрицы.

Матрица А имеет седловую точку Aij, если Aij является минимальным элементом в i-й строке и максимальным – в j-м столбце.

Запишем метод для ввода матрицы:

using System;

// Метод для ввода матрицы

int[,] InputMatrix()

{

Console.Write("Введите количество строк: ");

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

Console.Write("Введите количество столбцов: ");

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

int[,] matrix = new int[rows, cols];

Console.WriteLine("Введите элементы матрицы:");

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

{

for (int j = 0; j < cols; j++)

{

matrix[i, j] = int.Parse(Console.ReadLine());

}

}

return matrix;

}

Запишем метод для подсчета положительных элементов в строках с хотя бы двумя нулями:

void CountPositiveInRowsWithZeros(int[,] matrix)

{

int rows = matrix.GetLength(0);

int cols = matrix.GetLength(1);

int count = 0;

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

{

int zeroCount = 0;

int positiveCount = 0;

for (int j = 0; j < cols; j++)

{

if (matrix[i, j] == 0)

{

zeroCount++;

}

if (matrix[i, j] > 0)

{

positiveCount++;

}

}

if (zeroCount >= 2)

{

count += positiveCount;

}

}

Console.WriteLine($"Количество положительных элементов в строках с хотя бы двумя нулями: {count}");

}

Запишем метод для поиска седловых точек:

void FindSaddlePoints(int[,] matrix)

{

int rows = matrix.GetLength(0);

int cols = matrix.GetLength(1);

bool hasSaddlePoint = false;

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

{

for (int j = 0; j < cols; j++)

{

bool isMinInRow = true;

bool isMaxInCol = true;

// Проверка, является ли текущий элемент минималь-ным в строке

for (int k = 0; k < cols; k++)

{

if (matrix[i, k] < matrix[i, j])

{

isMinInRow = false;

break;

}

}

// Проверка, является ли текущий элемент макси-мальным в столбце

for (int k = 0; k < rows; k++)

{

if (matrix[k, j] > matrix[i, j])

{

isMaxInCol = false;

break;

}

}

// Если это седловая точка, выводим ее координаты

if (isMinInRow && isMaxInCol)

{

Console.WriteLine($"Седловая точка найдена в строке {i + 1}, столбце {j + 1} (значение: {matrix[i, j]})");

hasSaddlePoint = true;

}

}

}

if (!hasSaddlePoint)

{

Console.WriteLine("Седловых точек не найдено.");

}

}

Применим разработанные методы и решим задачу:

int[,] matrix = InputMatrix();

CountPositiveInRowsWithZeros(matrix);

FindSaddlePoints(matrix);

**Задание 3.**

Задан текст. Заменить самое длинное и самое короткое слово на заданную пользователем строку. Во всех остальных словах вхождение буквы 'О' увеличить на 1.

Запишем метод для предобработки текста:

string ProcessText(string text, string replacement)

{

string[] words = SplitTextIntoWords(text);

if (words.Length == 0)

{

return text; // Если нет слов, возвращаем исходный текст

}

string shortestWord = FindShortestWord(words);

string longestWord = FindLongestWord(words);

ReplaceWords(words, shortestWord, longestWord, replace - ment);

return JoinWords(words);

}

Запишем метод для разбиения текста на слова:

string[] SplitTextIntoWords(string text)

{

return text.Split(new char[] { ' ', '.', ',', '!', '?' }, StringSplitOptions.RemoveEmptyEntries);

}

Запишем метод для определения кратчайшего слова:

string FindShortestWord(string[] words)

{

string shortestWord = words[0];

foreach (string word in words)

{

if (word.Length < shortestWord.Length)

{

shortestWord = word;

}

}

return shortestWord;

}

Запишем метод для определения самого длинного слова:

string FindLongestWord(string[] words)

{

string longestWord = words[0];

foreach (string word in words)

{

if (word.Length > longestWord.Length)

{

longestWord = word;

}

}

return longestWord;

}

Запишем метод для замены самого короткого и самого длинного слова:

void ReplaceWords(string[] words, string shortestWord, string longestWord, string replacement)

{

for (int i = 0; i < words.Length; i++)

{

if (words[i] == shortestWord || words[i] == longestWord)

{

words[i] = replacement;

// Замена самого короткого и самого длинного слова

}

else

{

words[i] = IncreaseO(words[i]);

// Увеличение 'О'

}

}

}

Запишем метод для увеличения вхождений символа «O»:

string IncreaseO(string word)

{

// Увеличение вхождений 'О'

char[] chars = word.ToCharArray();

for (int i = 0; i < chars.Length; i++)

{

if (chars[i] == 'О' || chars[i] == 'о')

{

chars[i] = 'О'; // Увеличиваем 'О' на 1

// Добавляем еще одно 'О' после текущего

word = word.Insert(i + 1, "О");

i++; // Пропускаем добавленный символ

}

}

return word;

}

Запишем метод для соединения слов в предложение:

string JoinWords(string[] words)

{

return string.Join(" ", words);

}

Применим разработанные методы и решим задачу:

Console.WriteLine("Введите текст:");

string inputText = Console.ReadLine();

Console.WriteLine("Введите строку для замены:");

string replacementString = Console.ReadLine();

// Обработка текста

string resultText = ProcessText(inputText, replacementString);

// Вывод результата

Console.WriteLine("Результат:");

Console.WriteLine(resultText);

**Контрольные вопросы**

1. Что такое массив в C#?
2. Как объявить и инициализировать массив целых чисел заданного размера?
3. Как получить доступ к элементу массива?
4. Каковы основные различия между одномерными и многомерными массивами в C#?
5. Как можно перебрать все элементы массива с помощью цикла?
6. Как изменить размер массива после его создания?
7. Что произойдет, если попытаться получить доступ к элементу массива по индексу, который выходит за пределы его размера?
8. Как скопировать массив в другой массив?
9. Что такое строка в C#?
10. Как объявить и инициализировать строку?
11. Как получить длину строки?
12. Как извлечь подстроку из строки?
13. Как объединить две строки в одну?
14. Как проверить, содержит ли строка определенное слово или символ?
15. Как преобразовать строку в верхний регистр?
16. Как заменить часть строки на другую строку?
17. Как разбить строку на массив строк по заданному разделителю?
18. Как сравнить две строки на равенство?

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2017 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Браузер (Свободно распространяемое программное обеспечение).