Содержание

[1. Задание 1. Обработка одномерных и двумерных массивов 2](#_Toc517948213)

[1.1. Постановка задачи 2](#_Toc517948214)

[1.2. Тестовый пример 2](#_Toc517948215)

[1.3. Алгоритмы программы и процедуры №4 4](#_Toc517948216)

[1.4. Текст программы на языке C# 6](#_Toc517948217)

[1.5. Отладка программы 9](#_Toc517948218)

[1.6. Результаты работы программы 9](#_Toc517948219)

[1.7. Выводы 9](#_Toc517948220)

[2. Задание 2. Сортировка 10](#_Toc517948221)

[2.1. Постановка задачи 10](#_Toc517948222)

[2.2. Тестовый пример 10](#_Toc517948223)

[2.3. Алгоритм подпрограммы 11](#_Toc517948224)

[2.4. Текст программы на языке C# 12](#_Toc517948225)

[2.5. Отладка программы 13](#_Toc517948226)

[2.6. Результаты работы программы 13](#_Toc517948227)

[2.7. Выводы 13](#_Toc517948228)

[3. Задание 3. Рекурсия 14](#_Toc517948229)

[3.1. Постановка задачи 14](#_Toc517948230)

[3.2. Тестовый пример 14](#_Toc517948231)

[3.3. Алгоритм рекурсивной функции 14](#_Toc517948232)

[3.4. Текст программы на языке С# 16](#_Toc517948233)

[3.5. Отладка программы 17](#_Toc517948234)

[3.6. Результат работы программы 17](#_Toc517948235)

[3.7. Выводы 17](#_Toc517948236)

1. Задание 1. Обработка одномерных и двумерных массивов
   1. Постановка задачи

Вариант №5: разработать программу обработки матриц. Тип элементов матрицы – integer. Максимальная размерность матрицы 150x100 . Подпрограммы:

1. процедура ввода матрицы из текстового файла;
2. процедура вывода матрицы в окно терминала;
3. процедура вывода одномерного массива (вектора) в окно терминала;
4. процедура, заполняющая вектор, i-ый элемент которого равен нулю, если в i-ом столбце матрицы есть положительные элементы, и единице, в противном случае.

Головная программа должна демонстрировать применение вышеуказанных подпрограмм для двух матриц различной размерности.

* 1. Тестовый пример

Так как для тестирования программы изначально планировались матрицы, генерируемые случайно, заранее предугадать окончательный результат невозможно. Учитывая, что при выполнении программы в первую очередь файл с матрицей заполняется случайными числами, можно после выполнения программы просчитать результат для матрицы, хранящейся в файле, и сравнить его с полученным программой.

Полученная после очередного выполнения программы матрица:

.

Матрица такого же вида должна быть выведена процедурой программы.

Взяв для разложения на вектор 2 столбца матрицы, получим вектор вида:

.

Так как в первых двух столбцах есть положительные элементы, а в третьем их нет, то вектор должен будет иметь вид:

.

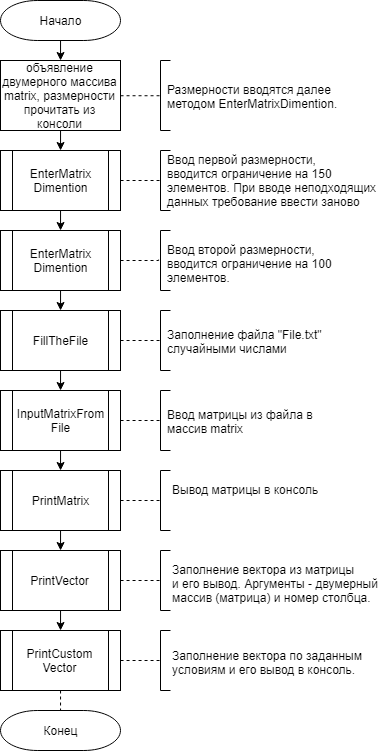
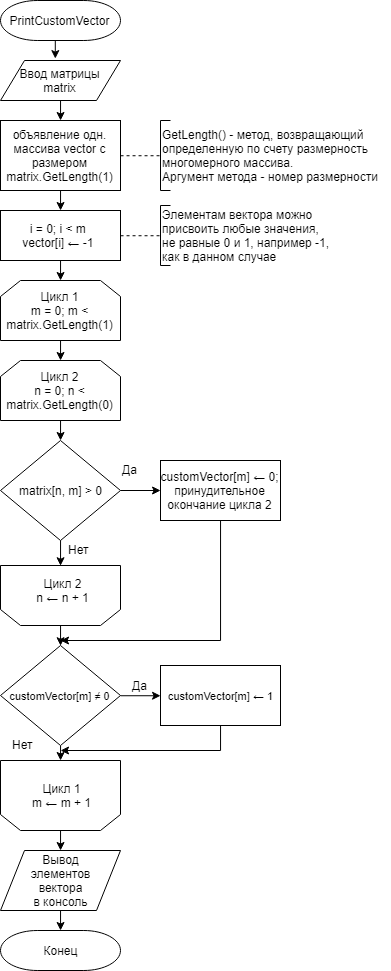
* 1. Алгоритмы программы и процедуры №4

Рис. 1 – Блок-схема головной программы

Рис. 2 – Блок-схема процедуры PrintCustomVector

* 1. Текст программы на языке C#

class Program

{

static void Main()

{

int[,] matrix = new int[EnterMatrixDimention('n', 150), EnterMatrixDimention('m', 100)];

FillTheFile("File.txt", matrix.GetLength(0), matrix.GetLength(1));

InputMatrixFromFile(ref matrix, "File.txt");

PrintMatrix(matrix);

Console.WriteLine("\n\n");

Console.WriteLine("Enter colomn number to get vector");

PrintVector(matrix, int.Parse(Console.ReadLine()));

Console.WriteLine("\n\n");

Console.WriteLine("Вектор с условием:");

PrintCustomVector(matrix);

Console.ReadLine();

}

//Ввод матрицы з файла

static void InputMatrixFromFile(ref int[,] matrix, string path)

{

//Открытие файла с путем path для чтения

StreamReader sr = new StreamReader(path);

for (int n = 0; n < matrix.GetLength(0); n++)

{

//Очередная строка преобразуется в массив строк,

//каждая из которых содержит строковое представление

//элементов матрицы

var stringLine = sr.ReadLine().Split(" ");

//Присваивание элементам очередной строки матрицы

//чисел, преобразованных из строк

for (int m = 0; m < matrix.GetLength(1); m++)

matrix[n, m] = int.Parse(stringLine[m]);

}

}

//Вывод матрицы в консоль

static void PrintMatrix(int[,] matrix)

{

for (int i = 0; i < matrix.GetLength(0); i++)

{

for (int j = 0; j < matrix.GetLength(1); j++)

{

Console.Write($"{matrix[i, j]} ");

}

Console.WriteLine();

}

}

//Ввод одной размерности матрирцы с консоли

static int EnterMatrixDimention(char c, int maxValue)

{

int dimention;

//Цикл будет бесконечно спрашивать размерность из

//консоли пока не получит подходящие данные

while (true)

{

//В блоке try проверяются условия, ограничивающие

//размерность

try

{

Console.WriteLine($"Введите {c}");

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

if (dimention < 1 || dimention > maxValue)

throw new Exception();

}

//Блок catch выполняется в случае ввода

//из консоли неправильных данных. Необходим для

//перехода в начало цикла

catch

{

Console.WriteLine("Something went wrong. Try again");

continue;

}

//Выход из метода в случае правильного ввода

return dimention;

}

}

//Вывод вектора, содержащего определенный столбец данной матрицы,

//в консоль

static void PrintVector(int[,] matrix, int m)

{

for (int i = 0; i < matrix.GetLength(0); i++)

Console.Write($"{matrix[i, m - 1]} ");

}

static void PrintCustomVector(int[,] matrix)

{

//Создание переменной размером в количество

//столбцов заданной матрицы

int[] customVector = new int[matrix.GetLength(1)];

//Заполнение вектора числами, не равными 0 и 1

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

customVector[i] = -1;

for (int m = 0; m < matrix.GetLength(1); m++)

{

for (int n = 0; n < matrix.GetLength(0); n++)

{

//Если i-й столбец содержит положительные числа,

//то i-му элементу приваивается 0, просматривание элементов

//прекращается

if (matrix[n, m] > 0)

{

customVector[m] = 0;

break;

}

}

//Если при просмотрении i-го стобца не было найдено

//положительных чисел, i-му элементу вектора присваивается 1

if (customVector[m] != 0) customVector[m] = 1;

}

//Вывод в консоль полученного вектора

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

Console.Write($"{customVector[i]} ");

}

//Заполнение файла случайными числами в форме матрицы через пробел

static void FillTheFile(string path, int n, int m)

{

//Удаление файла с данным именем

File.Delete(path);

//Создание файловой переменной sw и открытие файла

//с заданным именем для записи

StreamWriter sw = new StreamWriter(path);

//Создание объекта класса Random для возможности

//оперировать случайными числами

Random random = new Random();

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

{

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

{

//Random.Next - метод для генерации случайных чисел;

//аргумент метода - максимальное значение случайного числа

sw.Write($"{random.Next(-50, 150)} ");

}

sw.WriteLine();

}

//Dispose - метод для принудительной выгрузки буфера

sw.Dispose();

}

}

* 1. Отладка программы

При тестировании процедуры, заполняющей файл, возникла проблема, при которой файл после выполнения программы оказывался пустым. Она решилась добавлением в текст процедуры метода Dispose(), осуществляющего принудительную выгрузку буфера.

Кроме того, стандартные средства ввода языка C# предполагают ввод либо одного символа, либо целой строки. Поэтому стало необходимо разделить вводимую строку на подстроки, содержащие символьное представление чисел, чтобы в последствии преобразовать их к числам. Для этого использовался метод Split(), который разбивает строку на массив подстрок. Границей для разделения является аргумент метода, в данном случае символ пробела.

* 1. Результаты работы программы

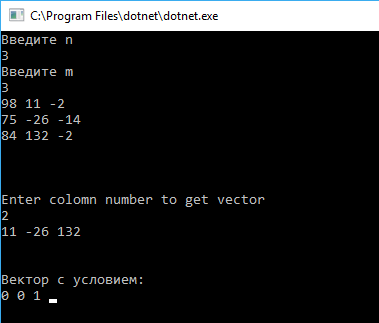


Рис. 3 – Результат работы программы

Результат работы программы совпадает с ожидаемым результатом, значит программа для работы с массивами разработана верно.

* 1. Выводы

Результатом выполнения задания 1 является освоение средств алгоритмических языков для работы с массивами. Кроме того, были изучены средства для работы с текстовыми строками языка C#.

1. Задание 2. Сортировка
   1. Постановка задачи

Добавить в программу, разработанную в соответствии с заданием 1, процедуру сортировки одномерного массива. Отсортировать одномерные массивы, формируемые в программе, используя заданный метод сортировки и тип упорядочивания.

* Метод сортировки: пузырек 1;
* Тип: по возрастанию.
  1. Тестовый пример

Запустим программу для генерации новой матрицы. Получим матрицу вида:

.

В результате сортировки по строкам должна получиться следующая матрица:

.

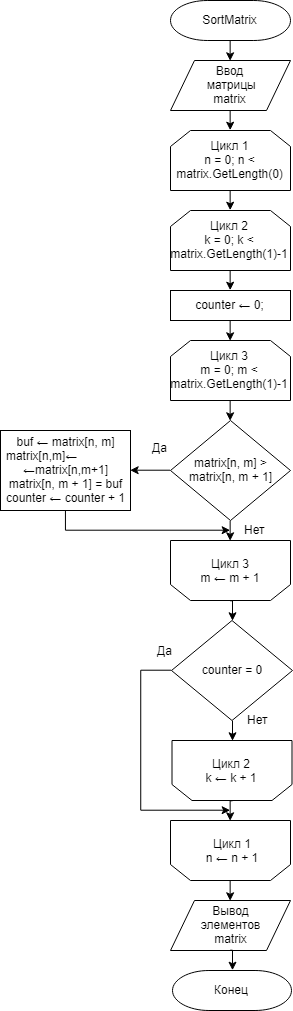
* 1. Алгоритм подпрограммы

Рис. 4 – Блок-схема процедуры SortMatrix

* 1. Текст программы на языке C#

class Task2

{

static void Main()

{

int[,] matrix = new int[EnterMatrixDimention('n', 150), EnterMatrixDimention('m', 100)];

FillTheFile("File.txt", matrix.GetLength(0), matrix.GetLength(1));

InputMatrixFromFile(ref matrix, "File.txt");

PrintMatrix(matrix);

Console.WriteLine("\n\n");

SortMatrix(ref matrix);

PrintMatrix(matrix);

Console.ReadLine();

}

static void SortMatrix(ref int[,] matrix)

{

for (int n = 0; n < matrix.GetLength(0); n++)

{

//k - номер этапа; кол-во этапов - n - 1

for (int k = 0; k < matrix.Length - 1; k++)

{

//Счетчик для досрочного завершения сортировки

int counter = 0;

//Просматриваем элементы с первого по последний,

//сравнивая пары элементов

for (int m = 0; m < matrix.GetLength(1) - 1; m++)

{

//Если левый элемент в паре больше чем правый,

//меняем их местами

if (matrix[n, m] > matrix[n, m + 1])

{

int buf = matrix[n, m];

matrix[n, m] = matrix[n, m + 1];

matrix[n, m + 1] = buf;

counter += 1;

}

}

//Если на этапе не было совершено обменов,

//то последовательность полностью отсортирована.

//В этом случае прекращаем сортировку

if (counter == 0) break;

}

}

}

}

* 1. Отладка программы

При написании программы обнаружилось, что если последовательность на каком-то этапе уже отсортирована, то можно досрочно прекратить сортировку. Для этого используется оператор break.

* 1. Результаты работы программы

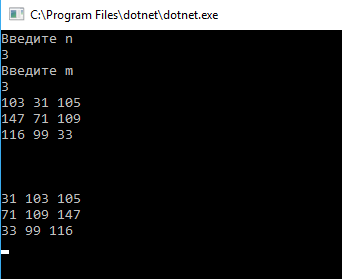


Рис. 5 – Результат работы программы

Результат работы программы совпадает с ожидаемым результатом, значит программа для сортировки массивов разработана верно.

* 1. Выводы

В ходе выполнения второго задания был изучен метод сортировки пузырьком, который был применен для сортировки строк матрицы.

1. Задание 3. Рекурсия
   1. Постановка задачи

Добавить в программу, разработанную в соответствии с заданиями 1 и 2, рекурсивную и итерационную функции обработки одномерного массива согласно варианту задания. Продемонстрировать работу функций.

* 1. Тестовый пример

Так как в случайно генерируемых матрицах вероятность появления нулевого элемента слишком низка для того, чтобы в полной мере продемонстрировать работоспособность процедуры, был описан массив вида:

Таким образом, количество нулевых элементов в последовательности равно 9, количество ненулевых – 6.

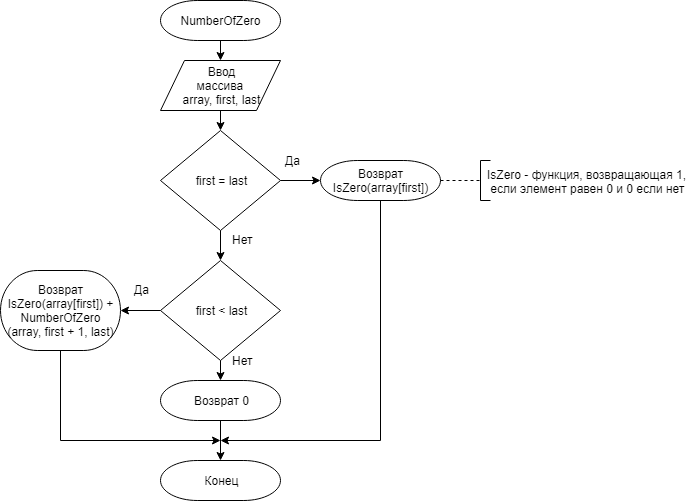
* 1. Алгоритм рекурсивной функции

Рис. 6 – Блок-схема рекурсивной функции

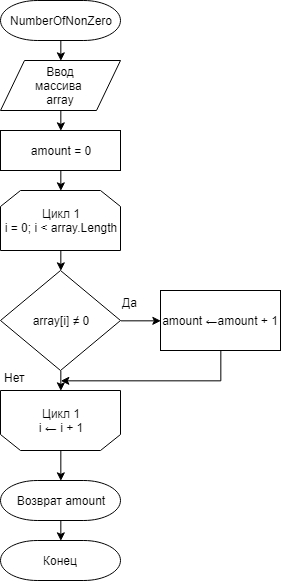


Рис. 7 – Блок-схема итеративной функции

* 1. Текст программы на языке С#

class Task3

{

static void Main()

{

int[] array = { 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1 };

foreach (int number in array) Console.Write($"{number} ");

Console.WriteLine();

Console.WriteLine($"Количество нулевых элементов(рекурсивно): {NumberOfZero(array, 0, array.Length - 1)}");

Console.WriteLine($"Количество ненулевых элементов(итерационно): {NumberOfNonZero(array)}");

Console.ReadLine();

}

static int NumberOfZero(int[] array, int first, int last)

{

//Нерекурсивная часть

if (first == last) return IsZero(array[first]);

//Рекурсивная часть

if (first < last) return IsZero(array[first]) + NumberOfZero(array, first + 1, last);

else return 0;

}

static int NumberOfNonZero(int[] array)

{

int amount = 0;

foreach (int number in array)

{

if (number != 0) amount += 1;

}

return amount;

}

static int IsZero(int number)

{

if (number == 0) return 1;

else return 0;

}

}

* 1. Отладка программы

При разработке рекурсивной функции появились множественные вложенные условные операторы, которые удалось вынести в отдельную функцию для увеличения читабельности текста программы.

* 1. Результат работы программы

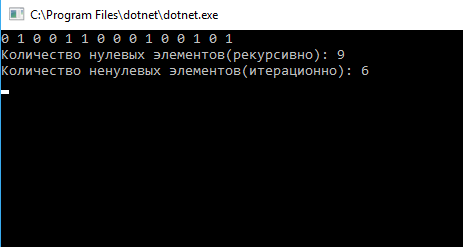


Рис. 8 – Результат работы программы

Результат работы программы совпадает с ожидаемым результатом, значит программа с рекурсивной функцией и итеративной процедурой работает верно.

* 1. Выводы

При выполнении третьего задания были изучены рекурсивные функции, а также методы обработки массивов с помощью рекурсивных и итеративных функций.