

# Лабораторная работа №3

по дисциплине «Программирование на Си»

## Обработка многомерных статических массивов

Кострицкий А. С., Ломовской И. В.

Москва — 2021 — TS2104091443

### Содержание

<b>1</b>	<b>Цель работы</b>	<b>1</b>
<b>2</b>	<b>Общее задание</b>	<b>1</b>
2.1	Задача №1	1
2.2	Задача №2	2
2.3	Задача №3	2
2.4	Задача №4	3
2.5	Задача №5	3
2.6	Задача №6	3
2.7	Примечания	4
<b>3</b>	<b>Взаимодействие с системой тестирования</b>	<b>5</b>

## 1 Цель работы

Целью лабораторной работы является знакомство студентов с обработкой матриц, хранящихся в виде двухмерных статических массивов.

Студент должен научиться описывать двухмерные статические массивы, обрабатывать их, передавать двухмерные массивы в функции.

## 2 Общее задание

В каждой задаче реализуйте программу, которая принимает у пользователя целочисленную матрицу и выполняет её обработку в соответствии с вариантом. Максимальное количество строк и столбцов матрицы равно десяти.

### 2.1 Задача №1

По матрице получить и вывести на экран одномерный массив, присвоив его  $k$ -тому элементу значение 1, если выполняется указанное условие, и значение 0 в иных случаях:

## Варианты

0. элементы  $k$ -го столбца упорядочены по убыванию;
1.  $k$ -ая строка матрицы симметрична;
2. в  $k$ -ом столбце чередуются положительные и отрицательные элементы;
3. элементы  $k$ -ой строки образуют монотонную последовательность<sup>1</sup>.

## 2.2 Задача №2

### Варианты

0. Вставить строку из чисел 100 после каждой строки, в которой количество элементов, начинающихся на заданную цифру, равно количеству элементов, заканчивающихся на эту же цифру. В ситуации, когда таких чисел не обнаружено вообще, строку из сотен не вставлять. Цифра вводится в виде числа после ввода матрицы.
1. Удалить строку и столбец, на пересечении которых расположен элемент матрицы, сумма цифр которого минимальна. При обнаружении нескольких подходящих элементов считать целевым первый при обходе по строкам.
2. Вставить строку из чисел  $-1$  перед каждой строкой, в которой есть хотя бы два элемента, сумма цифр каждого из которых нечётна.
3. Удалить из матрицы все столбцы, содержащие по крайней мере одно число, в записи которого встречается заданная цифра. Цифра вводится в виде числа после ввода матрицы.

## 2.3 Задача №3

### Варианты

Упорядочить строки<sup>2</sup> матрицы:

0. по возрастанию суммы их элементов;
1. по убыванию их наибольших элементов;
2. по возрастанию произведения их элементов;
3. по убыванию их наименьших элементов.

---

<sup>1</sup>Если в строке всего один элемент, то последовательности образовать нельзя, как следствие, нельзя образовать и монотонную последовательность.

<sup>2</sup>Обратите внимание, что в ключе не используется вся информация о каждой строке, следовательно, неустойчивые алгоритмы сортировки могут выдавать разные результаты на одних входных данных. На сервере во всех тестах в матрице гарантировано не будет двух строк, одинаковых с точки зрения ключа. Локально у себя Вы можете поступить одним из двух способов: либо использовать устойчивый алгоритм сортировки, либо нигде в тестовых данных не использовать матриц с двумя одинаковыми по ключу строками.

## 2.4 Задача №4

Для квадратной матрицы:

### Варианты

0. Найти и вывести на экран минимальное нечётное число, расположенное под главной диагональю.
1. Поменять местами элементы, расположенные в показанной в примере области. Первая строка меняется с последней, вторая — с предпоследней и т. д. Элементы, расположенные на главной и побочных диагоналях, включены в обмен.

$$\begin{pmatrix} \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} & \boxed{1} \\ 2 & \boxed{2} & 2 & \boxed{2} & 2 \\ 3 & 3 & \boxed{3} & 3 & 3 \\ 4 & \boxed{4} & \boxed{4} & \boxed{4} & 4 \\ \boxed{5} & \boxed{5} & \boxed{5} & \boxed{5} & \boxed{5} \end{pmatrix}$$

2. Найти максимальное число, которое заканчивается на цифру 5 и расположено под побочной диагональю.
3. Поменять местами элементы, расположенные в показанной на в примере области. Первый столбец меняется с последним, второй — с предпоследним и т. д. Элементы, расположенные на главной и побочных диагоналях, включены в обмен.

$$\begin{pmatrix} \boxed{1} & 1 & 1 & 1 & \boxed{1} \\ \boxed{2} & \boxed{2} & 2 & \boxed{2} & \boxed{2} \\ \boxed{3} & \boxed{3} & \boxed{3} & \boxed{3} & \boxed{3} \\ \boxed{4} & \boxed{4} & 4 & \boxed{4} & \boxed{4} \\ \boxed{5} & 5 & 5 & 5 & \boxed{5} \end{pmatrix}$$

## 2.5 Задача №5

### Варианты

0. Элементы матрицы, которые являются простыми числами, в порядке обхода матрицы по строкам поместить в одномерный массив, развернуть массив, и вернуть элементы из массива в матрицу в том же порядке, в котором они помещались в массив. Если в матрице нет простых чисел, считать ситуацию ошибочной.
1. Элементы матрицы, сумма цифр которых больше 10, в порядке обхода матрицы по строкам поместить в одномерный массив, циклически сдвинуть элементы этого массива влево на три позиции, и вернуть элементы из массива в матрицу в том же порядке, в котором они помещались в массив. Если в матрице нет чисел, сумма цифр которых больше 10, считать ситуацию ошибочной.

## 2.6 Задача №6

0. Приняв с клавиатуры число строк и столбцов матрицы, заполнить прямоугольную целочисленную матрицу «ходом быка» с начала: нечётные столбцы проходить сверху вниз, чётные — наоборот. Матрицу вывести на экран.

Пример:

$$\begin{pmatrix} 1 & 6 & 7 & 12 \\ 2 & 5 & 8 & 11 \\ 3 & 4 & 9 & 10 \end{pmatrix}$$

1. Приняв с клавиатуры число строк и столбцов матрицы, заполнить квадратную целочисленную матрицу по спирали по часовой стрелке. Матрицу вывести на экран.

Пример:

$$\begin{pmatrix} 1 & 2 & 3 \\ 8 & 9 & 4 \\ 7 & 6 & 5 \end{pmatrix}$$

2. Приняв с клавиатуры число строк и столбцов матрицы, заполнить прямоугольную целочисленную матрицу «ходом быка» с конца таким образом, чтобы максимальное число находилось в левом нижнем углу матрицы. Матрицу вывести на экран.

Пример:

$$\begin{pmatrix} 10 & 9 & 4 & 3 \\ 11 & 8 & 5 & 2 \\ 12 & 7 & 6 & 1 \end{pmatrix}$$
$$\begin{pmatrix} 13 & 12 & 7 & 6 & 1 \\ 14 & 11 & 8 & 5 & 2 \\ 15 & 10 & 9 & 4 & 3 \end{pmatrix}$$

3. Приняв с клавиатуры число строк и столбцов матрицы, заполнить квадратную целочисленную матрицу по спирали против часовой стрелки. Матрицу вывести на экран.

Пример:

$$\begin{pmatrix} 1 & 8 & 7 \\ 2 & 9 & 6 \\ 3 & 4 & 5 \end{pmatrix}$$

## 2.7 Примечания

1. *Статические массивы* следует отличать от *массивов переменной длины* (англ. *Variable Length Array, VLA*). Во избежание случайного использования последних при компиляции программы необходимо указывать ключ `-Wvla`.
2. Для реализации каждой из задач этой лабораторной работы необходимо выделить несколько осмысленных функций. Необходимо предусмотреть обработку ошибочных ситуаций.
3. При *вводе матрицы* сначала указывается количество строк и столбцов матрицы, затем вводятся сами элементы. Ввод неверного количества строк, столбцов, недостаточного количества самих элементов следует считать ошибочной ситуацией.
4. При *выводе матрицы* выводятся только её элементы построчно.
5. Ситуации, когда решение задачи не может быть получено, следует считать исключительными. Например, если нужно подсчитать количество чётных элементов массива, а таких элементов в массиве нет.

Помните, что в случае возникновения ошибочной ситуации программа должна не только выдавать соответствующее сообщение, но и возвращать ненулевой код возврата.

### 3 Взаимодействие с системой тестирования

1. Исходный код лабораторной работы размещается студентом в ветви `lab_LL`, а решение каждой из задач — в отдельной папке с названием вида `lab_LL_PP_CC`, где `LL` — номер лабораторной, `PP` — номер задачи, `CC` — вариант студента. Если дана общая задача без вариантов, решение следует сохранять в папке с названием вида `lab_LL_PP`.

Пример: решения восьми задач седьмого варианта пятой лабораторной размещаются в папках `lab_05_01_07`, `lab_05_02_07`, `lab_05_03_07`, ..., `lab_05_08_07`.

2. Исходный код должен соответствовать оглашённым в начале семестра правилам оформления.
3. Если для решения задачи студентом создаётся отдельный проект в IDE, разрешается поместить под версионный контроль файлы проекта, добавив перед этим необходимые маски в список игнорирования. Старайтесь добавлять маски общего вида. Для каждого проекта должны быть созданы, как минимум, два варианта сборки: **Debug** — с отладочной информацией, и **Release** — без отладочной информации.
4. Для каждой программы ещё до реализации студентом заготавливаются и помещаются под версионный контроль в подпапку `func_tests` функциональные тесты, демонстрирующие её работоспособность.

Позитивные входные данные следует располагать в файлах вида `pos_TT_in.txt`, выходные — в файлах вида `pos_TT_out.txt`, аргументы командной строки при наличии — в файлах вида `pos_TT_args.txt`, где `TT` — номер тестового случая.

Негативные входные данные следует располагать в файлах вида `neg_TT_in.txt`, выходные — в файлах вида `neg_TT_out.txt`, аргументы командной строки при наличии — в файлах вида `neg_TT_args.txt`, где `TT` — номер тестового случая.

Разрешается помещать под версионный контроль в подпапку `func_tests` сценарии автоматического прогона функциональных тестов. Если Вы используете при автоматическом прогоне функциональных тестов сравнение строк, не забудьте проверить используемые кодировки. Помните, что UTF-8 и UTF-8(BOM) — две разные кодировки.

Под версионный контроль в подпапку `func_tests` также помещается файл `readme.md` с описанием в свободной форме содержимого каждого из тестов. Вёрстка файла на языке Markdown обязательной не является, достаточно обычного текста.

Пример: восемь позитивных и шесть негативных функциональных тестов без дополнительных ключей командной строки должны размещаться в файлах `pos_01_in.txt`, `pos_01_out.txt`, ..., `neg_06_out.txt`. В файле `readme.md` при этом может содержаться следующая информация:

```

# Тесты для лабораторной работы №LL

## Входные данные
Целые a, b, c

## Выходные данные
Целые d, e

## Позитивные тесты:
- 01 - обычный тест;
- 02 - в качестве первого числа ноль;
...
- 08 - все три числа равны.

## Негативные тесты:
- 01 - вместо первого числа идёт буква;
- 02 - вместо второго числа идёт буква;
...
- 06 - вводятся слишком большие числа.

```

5. Если не указано обратное, успешность ввода должна контролироваться. При первом неверном вводе программа должна прекращать работу с ненулевым кодом возврата.
6. Вывод программы может содержать текстовые сообщения и числа. Если не указано обратное, тестовая система анализирует числа в потоке вывода, поэтому они могут быть использованы только для вывода результатов — использовать числа в информационных сообщениях запрещено.

Пример: сообщение «**Input point 1:**» будет неверно воспринято тестовой системой, а сообщения «**Input point A:**» или «**Input first point:**» — правильно.

Тестовая система вычлняет из потока вывода числа, обособленные пробельными символами.

Пример: сообщения «**a=1.043**» и «**a = 1.043.**» будут неверно восприняты тестовой системой, а сообщения «**a: 1.043**» или «**a = 1.043**» — правильно.

7. Если не указано обратное, числа двойной точности следует выводить, округляя до шестого знака после точки.