

Лабораторная работа №8

по дисциплине «Программирование на Си»

Обработка матриц

Кострицкий А. С., Ломовской И. В.

Москва — 2020 — TS2012211102

Содержание

Цель работы	1
Первый комплект вариантов	2
Общее задание	2
Примечания	2
Варианты удаления	2
Варианты добавления	2
Пример	3
Именованная папка с лабораторной	3
Второй комплект вариантов	3
Общее задание	3
Способы выделения памяти	4
Форматы файлов	4
Операции по вариантам	5
Именованная папка с лабораторной	5
Взаимодействие с системой тестирования	5
Памятка преподавателя	7

Цель работы

Целью работы является знакомство студентов с двумерными динамическими массивами. Студенты должны получить и закрепить на практике следующие знания и умения:

1. Выделение и освобождение памяти под двумерные динамические массивы.
2. Обработка матриц и текстовых файлов.
3. Организация корректной работы с ресурсами (динамически выделенная память, файловые дескрипторы).
4. Использование в программе аргументов командной строки.
5. Контроль правильности работы с динамической памятью с помощью специального ПО.

Первый комплект вариантов

Общее задание

Принять с клавиатуры

$$m \in \mathbb{N}, n \in \mathbb{N}, A \in M_{[m \times n]}(\mathbb{Z}),$$

$$p \in \mathbb{N}, q \in \mathbb{N}, B \in M_{[p \times q]}(\mathbb{Z}).$$

Удаляя строки или столбцы, в соответствии с вариантом привести матрицы A и B к квадратному виду A_1 и B_1 :

$$A_1 \in M_{[k \times k]}(\mathbb{Z}), \quad \text{where } k = \min(m, n),$$

$$B_1 \in M_{[s \times s]}(\mathbb{Z}), \quad \text{where } s = \min(p, q).$$

Обратите внимание: если у матрицы больше строк, чем столбцов, то удалять нужно только строки, иначе — только столбцы.

Добавляя в конец сначала строки, а потом столбцы, в соответствии с вариантом привести матрицы A_1 и B_1 к одному размеру A_2 и B_2 :

$$A_2, B_2 \in M_{[z \times z]}(\mathbb{Z}), \quad \text{where } z = \max(k, s).$$

Приняв с клавиатуры целые неотрицательные ρ и γ , вычислить и вывести на экран результат выражения

$$A_2^\rho B_2^\gamma = \underbrace{A_2 \cdot A_2 \cdot \dots \cdot A_2}_{\rho \text{ множителей}} \cdot \underbrace{B_2 \cdot B_2 \cdot \dots \cdot B_2}_{\gamma \text{ множителей}}.$$

Примечания

1. Считать любую квадратную матрицу в нулевой степени единичной.
2. Принять, что под вводом и выводом матриц подразумеваются «классические», насколько тут уместно это слово, построчные ввод и вывод оных.
3. Принять, что под перемножением матриц подразумевается «классическое», насколько тут уместно это слово, перемножение «строка на столбец».

Варианты удаления

Удалять строки или столбцы с *максимальным* (в вариантах 1, 2, 3, 4) или *минимальным* (в вариантах 5, 6, 7, 8) элементом в матрице. Если обнаружено несколько, считать целевым *максимум/минимум*, который был бы встречен *первым* (в вариантах 1, 3, 5, 7) или *последним* (в вариантах 2, 4, 6, 8) при обходе *по строкам* (в вариантах 1, 2, 5, 6) или *по столбцам* (в вариантах 3, 4, 7, 8).

Варианты добавления

Добавлять всегда сначала новые строки, потом — столбцы.

Добавлять в новые строки округлённые к нижнему целому *средние арифметические* (в вариантах 1, 2, 3, 4) или *средние геометрические модулей* (в вариантах 5, 6, 7, 8) элементов столбцов.

Добавлять в новые столбцы *максимумы* (в вариантах 1, 3, 5, 7) или *минимумы* (в вариантах 2, 4, 6, 8) по строкам.

Пример

Вариант №1: «Удалять строки или столбцы с *максимальным* элементом в матрице. Если обнаружено несколько, считать целевым *максимум*, который был бы встречен *первым* при обходе *по строкам*. Добавлять всегда сначала новые строки, потом — столбцы. Добавлять в новые строки округлённые к нижнему целому *средние арифметические* элементов столбцов. Добавлять в новые столбцы *максимумы* по строкам.»

В рамку обведены целевые максимумы, **красным** выделены удаляемые строки или столбцы.

$$\text{In: } n = 2, m = 3, A = \begin{pmatrix} 4 & \boxed{7} & 1 \\ 3 & \textcolor{red}{2} & 7 \end{pmatrix}, p = 5, q = 3, B = \begin{pmatrix} \textcolor{red}{0} & \textcolor{red}{2} & \boxed{9} \\ 1 & 2 & 2 \\ 3 & 3 & 3 \\ \boxed{8} & \textcolor{red}{8} & \textcolor{red}{2} \\ 1 & 8 & 3 \end{pmatrix}, \rho = 2, \gamma = 3;$$

$$k = 2, A_1 = \begin{pmatrix} 4 & 1 \\ 3 & 7 \end{pmatrix}, s = 3, B_1 = \begin{pmatrix} 1 & 2 & 2 \\ 3 & 3 & 3 \\ 1 & 8 & 3 \end{pmatrix};$$

$$z = 3, A_2 = \begin{pmatrix} 4 & 1 & \max(4, 1) \\ 3 & 7 & \max(3, 7) \\ \lfloor 3.5 \rfloor & \lfloor 4.0 \rfloor & \max(\lfloor 3.5 \rfloor, \lfloor 4.0 \rfloor) \end{pmatrix} = \begin{pmatrix} 4 & 1 & 4 \\ 3 & 7 & 7 \\ 3 & 4 & 4 \end{pmatrix}, B_2 = \begin{pmatrix} 1 & 2 & 2 \\ 3 & 3 & 3 \\ 1 & 8 & 3 \end{pmatrix};$$

$$\text{Out: } A_2^\rho B_2^\gamma = A_2^2 B_2^3 = A_2 \cdot A_2 \cdot B_2 \cdot B_2 \cdot B_2 = \begin{pmatrix} 15464 & 34369 & 22134 \\ 36567 & 81282 & 52327 \\ 22680 & 50421 & 32461 \end{pmatrix}.$$

Именование папки с лабораторной

Папка с лабораторной для этого комплекта вариантов носит имя `lab_08_VV_01`, где VV — номер варианта задания.

Второй комплект вариантов

Общее задание

Написать программу для работы с матрицами, которая реализует сложение матриц, умножение матриц и указанную ниже операцию.

Память под матрицы выделяется динамически.

Исходные матрицы читаются из файла, Результирующая матрица или число записываются в файл. Один файл содержит одну матрицу.

Тестирование выполняется с помощью сравнения полученного результата с ожидаемым. При этом нужно помнить, что сравнивать вещественные числа на равенство можно только с заданной точностью.

Имена файлов и выполняемая операция указывается через параметры командной строки. Формат запуска приложения должен быть следующим:

```
app.exe action mtr_1.txt [mtr_2.txt] res.txt
```

Возможные значения `action`:

1. **a** — сложение;
2. **m** — умножение;
3. **o** — операция по варианту (для неё `mtr_2.txt` не указывается).

Если операция по варианту «решение СЛАУ», то столбец свободных членов дописывается к матрице коэффициентов, и матрица размером $n \times (n + 1)$ помещается в файл `mtr_1.txt`. Столбец решений сохраняется в файле `res.txt` в виде матрицы.

Если операция по варианту «вычисление определителя», то файл `res.txt` содержит число — значение определителя.

Способы выделения памяти

1. *Массив указателей на строки*: при этом способе хранения каждая строка матрицы размещается в памяти отдельно от остальных. В памяти размещается, помимо данных, массив указателей на строки¹.
2. *Объединённый подход, способ 1*: при этом способе хранения данные хранятся единым блоком построчно. В памяти размещается, помимо данных, массив указателей на строки².
3. *Объединённый подход, способ 2*: при этом способе хранения вся матрица и массив указателей на строки размещаются единым блоком³.

Замечание

Крайне желательно продумать такую реализацию функций выделения памяти под матрицу и освобождения памяти из-под матрицы, чтобы можно было легко «переключаться» между разными способами выделения памяти.

Форматы файлов

1. *Простой формат*: количество строк и столбцов матрицы указывается в первой строке файла, остальные строки содержат сами элементы.

```
2 3
0 1 2
3 4 5
```

2. *Координатный формат*: количество строк и столбцов матрицы и количество ненулевых элементов указывается на первой строке матрицы. Остальные строки содержат тройки чисел: «номер строки», «номер столбца», «значение элемента». Указываются значения только элементов, отличных от нуля. Обратите внимание, что нумерация элементов в этом формате начинается с единицы.

```
2 2 3
1 1 1
1 2 2
2 1 3
```

¹Слайды 6 – 13 презентации.

²Слайды 14 – 21 презентации.

³Слайды 22 – 26 презентации.

Операции по вариантам

9. Решение СЛАУ методом Гаусса с выбором ведущего элемента по столбцу.
10. Решение СЛАУ методом Гаусса с выбором ведущего элемента по строке.
11. Решение СЛАУ методом Гаусса с выбором ведущего элемента по активной подматрице.
12. Вычисление определителя с помощью разложения по строке.
13. Вычисление определителя с помощью разложения по столбцу.
14. Вычисление обратной матрицы методом Гаусса. Обратная матрица ищется через решение системы $Ax = f$ с различными правыми частями. Правая часть последовательно пробегает значения столбцов e_j единичной матрицы E , при этом для каждой из них найденное решение системы $Ax = f$ образует j -ый столбец искомой обратной матрицы.
15. Вычисление обратной матрицы методом элементарных преобразований. К исходной матрице справа приписывается единичная матрица того же порядка: $A|E$. С помощью элементарных преобразований строк и столбцов левая «половина» приводится к единичной, совершаются одновременно точно такие же преобразования над правой матрицей.
16. Вычисление определителя методом Гаусса.

Замечание №1

На практике метод Гаусса не используется без выбора главного элемента. Поэтому если не оговорено специально и в формулировке задания указан метод Гаусса, считайте, что это метод Гаусса с выбором главного элемента по столбцу.

Замечание №2

Для сравнения чисел с плавающей точкой a и b при условии их близости к единице по модулю можно использовать неравенство $|a - b| < \varepsilon$, хотя возможна и оценка общего вида $|a - b| < \varepsilon \cdot \max(|a|, |b|)$.

Именование папки с лабораторной

Папка с лабораторной для этого комплекта вариантов носит имя `lab_08_VV_AB`, где `VV` — номер варианта задания, `A` — формат входных файлов, `B` — формат выходных файлов.

Примеры: `9_11` — вариант 9, входные файлы — простой формат, выходной файл — простой формат. `9_12` — вариант 9, входные файлы — простой формат, выходной файл — координатный формат. `9_21` — вариант 9, входные файлы — координатный формат, выходной файл — простой формат. `9_22` — вариант 9, входные файлы — координатный формат, выходной файл — координатный формат.

Способы выделения памяти в названии никак не указывается.

Взаимодействие с системой тестирования

1. Решение задачи оформляется студентом в виде многофайлового проекта. Для сборки проекта используется программа `make`, сценарий сборки `makefile` помещается под версионный контроль. В сценарии должны присутствовать цель `app.exe` — для сборки основной программы, и цель `unit_tests.exe` — для сборки модульных тестов.

2. В сценарии сборки рекомендуется обозначить, помимо прочих, следующие цели:

- (a) **unit** – сборка и прогон модульных тестов.
- (b) **func** – прогон функциональных тестов.
- (c) **clean** – очистка генерируемых файлов.

3. Исходный код лабораторной работы размещается студентом в ветви **lab_LL**, а решение каждой из задач — в отдельной папке с названием вида **lab_LL_CC_PP**, где **LL** — номер лабораторной, **CC** — вариант студента, **PP** — номер задачи.

Пример: решения восьми задач седьмого варианта пятой лабораторной размещаются в папках **lab_05_07_01**, **lab_05_07_02**, **lab_05_07_03**, ..., **lab_05_07_08**.

4. Исходный код должен соответствовать оглашённым в начале семестра правилам оформления.

5. 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.

6. 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.

```

7. Рекомендуется задавать следующую структуру проекта:

- (a) Все файлы исходных кодов хранятся в подпапке `src`.
- (b) Все файлы заголовков хранятся в подпапке `inc`.
- (c) Для каждого модуля создаётся и помещается в подпапку `unit_tests` один файл с модульными тестами, имя которого повторяет имя модуля с префиксом «`check_`». Основная программа модульного тестирования носит название «`check_main.c`».
- (d) Функциональные тесты оформляются в соответствии с предыдущими пунктами.
- (e) Сценарий сборки и конечные приложения генерируются в корне проекта.
- (f) Все остальные генерируемые файлы, в том числе объектные файлы и файлы статистики `gsov`, создаются в подпапке `out`.

Пример: папка с проектом для лабораторной работы, состоящего из текста программы и двух модулей, будет иметь следующий вид:

```
/lab_LL_CC_PP/  
  app.exe  
  makefile  
  unit_tests.exe  
  /inc/  
    unit_a.h  
    unit_b.h  
  /out/  
    main.o  
    unit_a.o  
    unit_b.o  
  /src/  
    main.c  
    unit_a.c  
    unit_b.c  
  /func_tests/  
    ...  
  /unit_tests/  
    check_main.c  
    check_unit_a.c  
    check_unit_b.c
```

8. Для каждой подпрограммы должны быть подготовлены модульные тесты с помощью фреймворка check, которые демонстрируют её работоспособность.
9. Все динамические ресурсы, которые уже были Вами успешно запрошены, должны быть высвобождены к моменту выхода из программы. Для контроля можно использовать, например, программы Dr. Memory или valgrind.
10. Успешность ввода должна контролироваться. При первом неверном вводе программа должна прекращать работу с ненулевым кодом возврата.

Обратите внимание, что даже в этом случае все динамические ресурсы, которые уже были Вами успешно запрошены, должны быть высвобождены.

11. Вывод Вашей программы может содержать текстовые сообщения и числа. Тестовая система анализирует только 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.

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

Памятка преподавателя

1. *Только для ЛР№8.* Совпадение структур и типов данных у студента и в задании не проверяется тестовой системой.