

## Требования к программам

1. Программа должна получать все параметры в качестве аргументов командной строки. Аргументы командной строки:

- 1)  $n$  – число строк и столбцов  $n \times n$  матрицы  $A$ ,
- 2)  $p$  – количество выводимых значений в матрице и векторах,
- 3)  $k$  – задает номер формулы для инициализации матрицы  $A$ , должен быть равен 0 при вводе матрицы  $A$  из файла,
- 4) `filename` – имя файла, откуда надо прочитать матрицу  $A$ . Этот аргумент **отсутствует**, если  $k \neq 0$ ,

Например, запуск

```
./a.out 4 4 0 a.txt
```

означает, что матрицу  $A$   $4 \times 4$  надо прочитать из файла `a.txt`, и выводить не более 4-х строк и столбцов матрицы; а запуск

```
./a.out 2000 6 1
```

означает, что матрицу  $A$   $2000 \times 2000$  надо инициализировать по формуле номер 1 и выводить не более 6-ти строк и столбцов матрицы.

2. В задачах, где требуется правая часть  $b$ , он строится после инициализации матрицы  $A = (a_{i,j})_{i,j=1,\dots,n}$  по формуле:

$$b = (b_i)_{i=1,\dots,n}, \quad b_i = \sum_{k=0}^{(n-1)/2} a_{i,2k+1}$$

Инициализация должна быть оформлена в виде подпрограммы, вызываемой из функции `main`.

3. Ввод матрицы должен быть оформлен в виде подпрограммы, находящейся в отдельном файле.

4. Ввод матрицы из файла. В указанном файле находится матрица в формате:

$$\begin{array}{ccc} a_{1,1} & \dots & a_{1,n} \\ a_{2,1} & \dots & a_{2,n} \\ \dots & \dots & \dots \\ a_{n,1} & \dots & a_{n,n} \end{array}$$

где  $n$  – указанный размер матрицы,  $A = (a_{i,j})$  – матрица. Программа должна выводить сообщение об ошибке, если указанный файл не может быть прочитан, содержит меньшее количество данных или данные неверного формата.

5. Ввод матрицы и правой части по формуле. Элемент  $a_{i,j}$  матрицы  $A$  размера  $n \times n$  полагается равным

$$a_{i,j} = f(k, n, i, j), \quad i = 1, \dots, n, \quad j = 1, \dots, n,$$

где  $f(k, n, i, j)$  - функция, которая возвращает значение  $(i, j)$ -го элемента  $n \times n$  матрицы по формуле номер  $k$  (аргумент командной строки). Функция  $f(k, n, i, j)$  должна быть оформлена в виде отдельной подпрограммы.

$$f(k, n, i, j) = \begin{cases} n - \max\{i, j\} + 1 & \text{при } k = 1 \\ \max\{i, j\} & \text{при } k = 2 \\ |i - j| & \text{при } k = 3 \\ \frac{1}{i + j - 1} & \text{при } k = 4 \end{cases}$$

6. Решение должно быть оформлено в виде функции, находящейся в отдельном файле и получающей в качестве аргументов

- (a) размерность  $n$  матрицы  $A$ ,
- (b) матрицу  $A$ ,
- (c) правую часть  $b$  (если стоит задача решить линейную систему)
- (d) вектор  $x$ , в который будет помещено решение системы, если стоит задача решить линейную систему, или матрицу  $X$ , в которую будет помещена обратная матрица, если стоит задача обратить матрицу,
- (e) дополнительные вектора, если алгоритму требуется дополнительная память.

Получать в этой функции дополнительную информацию извне через глобальные переменные и т.п. запрещается.

7. Функция, реализующая задачу, возвращает ненулевое значение, если алгоритм решения неприменим к поданной на вход матрице  $A$ .
8. Функция, реализующая задачу, **не должна выделять или использовать дополнительную память**.
9. **Сложность работы функции**, реализующая задачу, не должна превышать  $O(n^3)$ .
10. Суммарный объем оперативной памяти, требуемой программе, не должен превышать:
- при вычислении решения системы:  $n^2 + O(n)$ ,
  - при вычислении обратной матрицы:  $2n^2 + O(n)$ .

Для выполнения этого требования после завершения алгоритма решения (нахождения обратной матрицы) вызывается подпрограмма инициализации матрицы (из файла или по формуле) и вычисления вектора  $b$  (в задачах решения линейной системы).

11. Программа должна содержать подпрограмму вывода на экран прямоугольной матрицы  $m \times n$  матрицы. Эта подпрограмма используется для вывода исходной  $n \times n$  матрицы после ее инициализации, а также для вывода на экран результата работы программы. Подпрограмма выводит на экран не более, чем  $p$  строк и столбцов  $m \times n$  матрицы, где  $p$  – параметр этой подпрограммы (аргумент командной строки). Каждая строка матрицы должна печататься на новой строке, каждый элемент матрицы выводится в строке по формату "%10.3e" (один пробел между элементами и экспоненциальный формат %10.3e).

12. Результатами работы программы являются 3 элемента:

- Собственно вектор решения  $x$  (в задачах нахождения решения линейной системы) или обратная матрица  $A^{-1}$  (в задачах нахождения обратной матрицы).
- Два вещественных числа  $r_1$  и  $r_2$ , вычисляемых после вызова задачи:

– В задачах нахождения решения линейной системы

$$r_1 = \|Ax - b\|_1 / \|b\|_1, \quad r_2 = \sum_{i=1}^n |x_i - (i \bmod 2)| / \sum_{i=1}^n (i \bmod 2) \quad \text{где } \|y\|_1 = \sum_{i=1}^n |y_i|.$$

– В задачах нахождения обратной матрицы

$$r_1 = \begin{cases} \|AA^{-1} - E\|_1, & N \leq 4000, \\ 0, & N > 4000 \end{cases} \quad r_2 = \begin{cases} \|A^{-1}A - E\|_1, & N \leq 4000, \\ 0, & N > 4000 \end{cases}$$

где

$$\|Y\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |y_{ij}|, \quad Y = (y_{ij})_{i,j=1,\dots,n}$$

Вычисление  $r_1$  и  $r_2$  должно быть оформлено в виде подпрограммы, вызываемой из функции `main`. Эта подпрограмма **не должна выделять или использовать дополнительную память**.

13. Вывод результата работы функции в функции `main` должен производиться по формату:

- Непосредственно вывод вектора решения  $x$  или обратной матрицы  $A^{-1}$ :
  - в задачах нахождения решения линейной системы вывод вектора решения  $x$  производится вызовом подпрограммы печати матрицы (см. пункт 11) размера  $1 \times n$  (т.е. в строку и **по указанному там формату**)
  - в задачах нахождения обратной матрицы вывод обратной матрицы  $A^{-1}$  производится вызовом подпрограммы печати матрицы (см. пункт 11) размера  $n \times n$  (**по указанному там формату**)
- Отчет о результате и времени работы:

```
printf (
    "%s : Task = %d Res1 = %e Res2 = %e Elapsed = %.2f K = %d N = %d\n",
    argv[0], task, r1, r2, t, k, n);
```

где

- `argv[0]` – первый аргумент командной строки (имя образа программы),
- `task` – номер задачи,
- `r1` – вычисленное значение  $r_1$  (см. пункт 12),
- `r2` – вычисленное значение  $r_2$  (см. пункт 12),
- `t` – время работы функции, реализующей решение этой задачи,
- `k, n` – аргументы командной строки.

**Вывод должен производиться в точности в таком формате**, чтобы можно было автоматизировать обработку запуска многих тестов.

## Задачи

1. Метод Гаусса решения линейной системы.
2. Метод Гаусса нахождения обратной матрицы.
3. LU-разложение для решения линейной системы.
4. LU-разложение для нахождения обратной матрицы.
5. Метод Холецкого решения линейной системы с симметричной матрицей.
6. Метод Холецкого нахождения обратной матрицы для симметричной матрицы.
7. Метод Жордана решения линейной системы.
8. Метод Жордана нахождения обратной матрицы.
9. Метод Гаусса решения линейной системы с выбором главного элемента по столбцу.
10. Метод Гаусса решения линейной системы с выбором главного элемента по строке.
11. Метод Гаусса решения линейной системы с выбором главного элемента по всей матрице.
12. Метод Гаусса нахождения обратной матрицы с выбором главного элемента по столбцу.
13. Метод Гаусса нахождения обратной матрицы с выбором главного элемента по строке.
14. Метод Гаусса нахождения обратной матрицы с выбором главного элемента по всей матрице.
15. Метод Жордана решения линейной системы с выбором главного элемента по столбцу.
16. Метод Жордана решения линейной системы с выбором главного элемента по строке.
17. Метод Жордана решения линейной системы с выбором главного элемента по всей матрице.
18. Метод Жордана нахождения обратной матрицы с выбором главного элемента по столбцу.
19. Метод Жордана нахождения обратной матрицы с выбором главного элемента по строке.
20. Метод Жордана нахождения обратной матрицы с выбором главного элемента по всей матрице.