ТРЕБОВАНИЯ К ПРОГРАММАМ
(многопоточная версия)

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

   1) n – размерность матрицы,
   2) p – количество использованных потоков,
   3) r – количество выводимых значений в матрице,
   4) s – задает номер формулы для инициализации матрицы, должен быть равен 0 при вводе матрицы из файла
   5) `filename` – имя файла, откуда надо прочитать матрицу. Этот аргумент **отсутствует**, если $s! = 0$.

   Например, запуск

   ```
   ./a.out 4 3 2 4 0 a.txt
   ```

   означает, что матрицу `4x4` надо прочитать из файла `a.txt`, использовать 2 потока и выводить не более 4-х строк и столбцов матрицы, а запуск

   ```
   ./a.out 2000 90 8 10 1
   ```

   означает, что матрицу `2000x2000` надо инициализировать по формуле номер 1, использовать 8 потоков и выводить не более 10-ти строк и столбцов матрицы.

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

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

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 полагается равным

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

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

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

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

   (a) размерность n матрицы A,

   (b) номер потока k, где работает этот "экземпляр" функции,

   (c) число потоков p,

   (d) матрицу A,

   (e) правую часть b (если стоит задача решить линейную систему)

   (f) 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,

   (g) дополнительные вектора, если алгоритму требуется дополнительная память.

   Получать в этой подпрограмме дополнительную информацию извне через глобальные переменные, включаемые файлы и т.п. запрещается.

7. Программа должна содержать подпрограмму вывода на экран прямоугольной матрицы $l \times n$ матрицы. Эта подпрограмма используется для вывода исходной $n \times n$ матрицы после ее инициализации, а также для вывода на экран решения системы ($1 \times n$ матрицы) или обратной $n \times n$ матрицы, если стоит задача обратить матрицу. Подпрограмма выводит на экран не более, чем r строк и столбцов $l \times n$ матрицы, где r – 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`).

8. Программа должна содержать подпрограмму вычисления нормы невязки, т.е.

   - при вычислении решения системы: $\|Ax-b\|/\|b\|$,
   - при вычислении обратной матрицы: $\|AA^{-1}-E\|$

   и выводить эту норму невязки на экран (в экспоненциальном формате). Вычисление невязки должно производиться всеми потоками параллельно.

9. Для задачи решения линейной системы программа должна выводить норму погрешности, т.е. норму разности между полученным приближенным решением и точным решением $(1,0,1,0,1,\dots)$.

10. Программа должна выводить на экран время, затраченное на решение системы (обращение матрицы, если стоит задача обратить матрицу), а также время, затраченное на вычисление невязки. Допустимо представлять время в сотых долях секунды, не преобразовывая в обычный формат чч.мм.сс:тт. Должно выводится астрономическое время время и процессорное время для каждого потока.

11. Программа должна выводить краткую информацию о запуске **в точности в указанном ниже формате**:

```
printf("%s : residual = %e elapsed = %.2f s = %d n = %d m = %d p = %d\n",
       argv[0], residual, elapsed, s, n, m, p);
```

   где:

   - `argv[0]` – первый аргумент командной строки (имя образа программы),
   - `residual` – невязка,

- `elapsed` – астрономическое время, затраченное на решение системы (нахождение обратной матрицы),
- `s`, `n`, `m`, `p` – аргументы командной строки.

Выводить требуется в точности так, чтобы этот текст можно было найти поиском по протоколу работы программы.

12. 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).

13. Время работы программы не должно превышать $O(n^3)$,

МЕТОДЫ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

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.
21. Метод вращений решения линейной системы.
22. Метод вращений нахождения обратной матрицы с использованием присоединенной матрицы.
23. Метод вращений нахождения обратной матрицы с использованием QR-разложения.
24. Метод отражений решения линейной системы.
25. Метод отражений нахождения обратной матрицы с использованием присоединенной матрицы.
26. Метод отражений нахождения обратной матрицы с использованием QR-разложения.