Вариант 22

Из входного потока вводится прямоугольная сильно разреженная матрица целых чисел [a_{ij}], $i = 1, \ldots, m$, $j = 1, \ldots, n$. Значения m и n заранее не известны и вводятся из входного потока.

Сформировать новую матрицу, поместив в ее i-ую строку элементы из i-ой строки исходной матрицы, размещенные между первым элементом > 0 и первым элементом < 0 (порядок расположения элементов произволен). Если в строке нет элемента, удовлетворяющего какому-либо одному условию, то выбираются элементы до конца строки исходной матрицы. Если строка исходной матрицы содержит только нули (т.е. не выполняются оба условия), то в результирующую матрицу новая строка не включается.

Исходную и полученную матрицы вывести в выходной поток с необходимыми комментариями.

Требования:

1) Выбранная структура матрицы должна учитывать специфику разреженной матрицы (оптимальность по памяти), и по сложности алгоритмов из задания (оптимальность по времени обработки). Допустимы порядковые отклонения от оптимального решения в случае обоснования решения - "trade-off" анализ (обязательно)

2) Проект на git'e. (обязательно)

3) Проверка корректности ввода (работа через потоки C++) (обязательно)

4) Статический анализ кода, встроенный инструментарий в IDE (пр. VS2019: Analyze->Run Code Analysis, см. также Project -> Properties -> Configuration Properties -> Code Analysis -> Microsoft -> Active Rules) или внешние инструменты (Sonarqube + extensions, Clang Static Analyzer и д.р.) (обязательно знакомство с инструментом, все исправлять не надо, т.к. некоторые вещи просто нельзя исправить без использования STL)

5) Динамический анализ на утечки памяти, встроенный инструментарий в IDE / библиотеки (Пр., VS2019) или внешние инструменты (valgrind, Deleaker и т.п.). (обязательно)

6) Не "кривой", не избыточный, поддерживаемый и расширяемый код (разумная декомпозиция, DRY, корректное использование заголовочных файлов и т.п.) (обязательно)

7) Использование ООП возможно (но при полностью корректной реализации, т.е. наличие необходимых конструкторов / деструкторов, перегрузка операторов, корректная сигнатура методов и т.п.), большинству не рекомендуется (дождитесь задач №2 и, особенно, №3).

8) Использование контейнеров из STL возможно только для "профессионалов" в C++, подавляющему большинству не рекомендуется (дождитесь задачи №4).

9) Стандарт языка C++17 (рекомендуется), C++20 (при наличии). Допустим C++11 или C++14 (если почему-то нет C++17)