

Общие требования:

- 1) Наличие интерактивного диалогового интерфейса для проверки корректности разработанной программы.
- 2) Корректное завершение программы, как в случае штатного выхода, так и в случае невосстановимых ошибок (без утечек и без использования функций мгновенного завершения программы `exit`, `abort`, `std::terminate` и пр.).
- 3) Проверка корректности ввода (работа только через потоки C++, т.е. без `scanf/printf/fscanf/fprintf`). В случаях ошибок формата ввода запрос повторного ввода данных. В случае невосстановимых ошибок ввода-вывода завершение программы.
- 4) Использование средств языка C++ для работы с динамической памятью – операторов `new` и `delete` (`malloc`, `calloc`, `realloc`, `free` запрещены).
- 5) Использование исключений для обработки ошибочных ситуаций (вместо кодов возврата).
- 6) Предпочтительно использование стандартных библиотек и функций языка C++ вместо библиотек и функций языка C (`std::copy` вместо `memcpy`, `std::abs` вместо `abs`, `cstring` вместо `string.h` и т.д.).
- 7) Логичная и удобная структура проекта, где каждая единица (файл/библиотека) обладает своей единой зоной ответственности (каждый класс в своих файлах `.h` и `.cpp`, диалоговые функции и `main` в своих).
- 8) Наличие средств автосборки проекта (желательно CMake, qmake и прочие, работающие “поверх” Makefile; использование самописного Makefile нежелательно, но допустимо).
- 9) Не "кривой", не избыточный, поддерживаемый и расширяемый код (разумная декомпозиция, DRY, корректное использование заголовочных файлов и т.п.).
- 10) Стандарт языка C++20 (рекомендуется). Допустим C++17 (если почему-то нет C++20).

Требования задачи:

- 1) Выбранная структура (класс) матрицы и операции с ней должны учитывать специфику симметричной (т.е. для любых i и j , $matrix[i][j] == matrix[j][i]$) матрицы. Необходимо обеспечить симметричность исходной матрицы и не давать пользователю возможности нарушать это свойство. Выходная матрица может быть как симметричной, так и нет, в зависимости от задания.
- 2) 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).
- 3) Использование контейнеров из STL возможно только для "профессионалов" в C++, подавляющему большинству не рекомендуется (дождитесь задачи №3).