Техническое задание: Система логирования на C++

Цель: Создание гибкой и многопоточной системы логирования на языке программирования C++.

Общие требования:

- Проектирование на основе ООП и паттернов (в частности, рекомендуется использование паттерна синглтон на стадиях 3 и 4).

Требования уровня 3:

1. **Уровни логирования:**
   - Определение пяти уровней: TRACE, DEBUG, INFO, WARNING, ERROR.
   - *Пример*: Если выбран уровень INFO, сообщения уровней TRACE и DEBUG игнорируются.
2. **Настройка мест вывода:**
   - Возможность выбора мест вывода: консоль, файл или оба варианта.
     Пример: Пользователь может выбрать вывод ТОЛЬКО в консоль, одновременно в консоль и файл или ТОЛЬКО в файл.
3. **Имя файла лога:**
   - Стандартное имя: [название_проекта].log.
   - Функциональность для изменения стандартного имени.
4. **Форматирование лога:**
   - В каждой записи присутствуют: дата/время, уровень логирования, текст сообщения.
     Пример: **2023-09-22 12:00:00 | INFO -> User logged in**.
5. **Макросы:**
   - Простое добавление сообщений в лог.
     Пример: Вместо **Log::getInstance()->write(Log::Severity::ERROR, "Error occurred")** можно использовать **LOGE("Error occurred")**.

Требования уровня 4:

1. **Документация:**
   - Оформление подробной документации по библиотеке в стиле *JAVADOC*.
2. **Расширенный формат:**
   - Добавление информации о директории, файле и номере строки исходника.
   - *Пример*: **2023-09-22 12:05:00 | DEBUG | src/main.cpp:45 -> Variable x initialized**.
3. **Работа с несколькими файлами:**
   - Поддержка записи в несколько лог-файлов одновременно.
   - Возможность выбора "таргета" или целевого файла для каждой записи.
4. **Уникальные имена файлов:**
   - Генерация уникального имени для каждого нового запуска программы.
   - *Пример*: Первый запуск создает файл **log_2023-09-22_12-00-00**, второй – **log_2023-09-22_12-05-00**.

Требования уровня 5:

1. **Гибкая передача параметров:**
    - Поддержка различных типов данных: числа, строки и т.д.
    - Самих параметров может быть сразу несколько, а не только текстовое сообщение или одно число.
    - *Пример*: **LOGE("Error in function", functionName, "with code", errorCode)**.
2. **UTF-8:**
    - Поддержка символов в формате UTF-8, включая смайлики.
3. **Пользовательские шаблоны:**
    - Настраиваемый шаблон для каждой записи.
    - *Пример*: Шаблон **{t} | {L} -> {m}** дает результат **2023-09-22 12:10:00 | INFO -> User logged out**.

Дополнительные требования для "отлично":

1. **Многопоточность:**
    - Асинхронное и многопоточное логирование.
    - Отчет о механизме многопоточности с визуальной схемой.

P.S.: При проектировании использовать шаблоны C++ для универсальности методов и классов.