Написать консольную программу, которая имеет следующий функционал:
Основной поток обрабатывает пользовательский клавиатурный ввод и управляет работой других потоков.
Второй поток (ведущий) через случайные промежутки времени генерирует события, которые должны протоколироваться третьим потоком.
Событие представляет собой структуру Event, содержащую поля:

- текущие дата и время,
- идентификатор события,
- 3 параметра типа int,
- строка-примечание длиной до 256 символов.

Передача событий между потоками должна осуществляться посредством очереди (можно реализовать с помощью анонимных пайпов).
Протоколирование должно быть многоуровневым с возможностью переключения режимов «на лету».
Объект протоколировщика любого уровня представляет собой потомок абстрактного класса Logger, который содержит прототип метода протоколирования и фабричный метод для порождения экземпляров классов-наследников.

```
class Logger
{
protected:
        Logger(char* fileName);
        ~Logger();
public:
        static Logger* GetLogger(int level);
        virtual Logger* Write(Event event);
}
```

- Нулевой уровень протоколирования записывает информацию о текущем времени и идентификаторе события.
- Первый уровень записывает информацию о дате и времени, идентификаторе события и первом параметре
- Второй уровень записывает полную информацию о событии

Кроме того, данные должны быть записаны в файловую базу данных SQLite.

Команды пользовательского ввода:
date – печать текущей даты
time – печать текущего времени
exit – завершить работу программы
faster – уменьшить интервал между порождениями событий
slower – увеличить интервал между порождением событий
pause – приостановить генерацию событий
resume – продолжить генерацию событий
level 0 – установить нулевой уровень протоколирования
level 1 – установить первый уровень протоколирования
level 2 – установить второй уровень протоколирования
stat – вывести количество произошедших событий

В задании реализовать сопоставление команды и её обработчика посредством std::map<std::string, {routine}>
Обработчик должен принимать в качестве параметров остаток строки, идущий после серии пробельных символов, как const char*. Если строка не содержит параметров, в обработчик следует передавать NULL.
Организовать вывод сообщений об ошибках при передаче некорректных аргументов и при запросе некорректной команды через поток вывода stderr.

Проект должен быть кроссплатформенным (Windows, Linux) с использованием CMake, а также размещённым в системе контроля версий Git (можно использовать github или иной публичный репозиторий).