Messages queue

C++ Software Developer

Qualification Test

Messages queue

Task:

You are asked to implement an asynchronous message exchange queue between ‘Writer’ (aka ‘Producer’) thread and “Reader” (aka ‘Consumer’) thread. This queue is supposed to be used for temporary traffic peaks mitigation (e.g. 'Reader' is consuming messages with a constant rate, while ‘Writer’ produces messages randomly)

Requirements:

1. Queue must be of fixed size (specified during construction)
2. Queue ‘pop’ operation must be blocking: returns a message or blocks waiting for it
3. Queue ‘push’ operation must be non-blocking: in case message addition is not possible, returns an error.
4. Every message sent by a ‘writer’ thread must be consumed by a ‘reader’ thread in FIFO order.
5. Class code should be written in ‘production’ style. That is the class should be reusable, containing all the necessary attributes, and being ready to be plugged into application.
6. No external libraries allowed. You can use C++ and STL only.

Please avoid over-complication. The implementation should be as simple as possible, conforming the requirements above. If you have questions or need requirements clarification, feel free to ask or make your own additional requirements (keeping the original requirements intact).

Bonus:

1. Implement ‘close’ function, that makes further pop/push impossible (update the interface and implementation of pop/push functions accordingly). Close can be entered by any thread.
2. Make ‘push’ enterable by several ‘writer’ threads, ‘pop’ enterable by several ‘reader’ threads.
3. Implement get (PREDICATE) method, which accepts predicate, and returns first matching element from the queue.
4. Implement blocking ‘push’ method: in case message addition is not possible, wait until it possible
5. Implement non-blocking ‘pop’ method: in case message is not available, return an error

NOTE: this task is not expected to take more than 2-3 hours of your time, so don’t spend time on environment or build setup (if you don’t have one configured already): just focus on sources and interfaces instead.

Required artifacts:

Please provide
- queue source code and a simple sample app demonstrating the queue functionality. It is desired for that source code to be buildable, but that is not required
- Detailed description of all functional methods of all implemented classes, including thread safety guarantees.document