

Interview Exam: SequenceSearcher Code

Interviewer

March 13, 2025

Instructions

Answer the following questions to the best of your ability. Provide clear and concise explanations. You may refer to the code provided below.

Code Reference

```
1 #include <iostream>
2 #include <vector>
3 #include <thread>
4 #include <mutex>
5 #include <queue>
6 #include <algorithm>
7
8 class SequenceSearcher {
9 public:
10     using ProcessingCallback = std::function<void(std::vector
        <uint8_t>&)>;
11
12     SequenceSearcher() {
13         std::cout << "[SequenceSearcher] Constructor." << std
        ::endl;
14     }
15
16     ~SequenceSearcher() {
17         std::cout << "[SequenceSearcher] Destructor." << std
        ::endl;
18     }
19
20     void start() {
```

```

21         isRunning = true;
22         processingThread = std::thread([this]() {
processQueueDataAndInvokeCallback(); });
23     }
24
25     void stop() {
26         isRunning = false;
27         if (processingThread.joinable()) {
28             processingThread.join();
29             std::cout << "[SequenceSearcher] Thread joined."
<< std::endl;
30         } else {
31             std::cout << "[WARNING][SequenceSearcher] Thread
not joinable!" << std::endl;
32         }
33     }
34
35     bool processVector(std::vector<uint8_t>& byteVector) {
36         const std::vector<uint8_t> TARGET_SEQUENCE = { 0x01,
0x02, 0x03 };
37         bool isSequenceFound = false;
38
39         auto it = std::search(
40             byteVector.begin(), byteVector.end(),
TARGET_SEQUENCE.begin(), TARGET_SEQUENCE.end()
41         );
42
43         if (it != byteVector.end()) {
44             isSequenceFound = true;
45             std::cout << "[SequenceSearcher] Sequence found."
<< std::endl;
46         } else {
47             std::cout << "[SequenceSearcher] Sequence not
found." << std::endl;
48         }
49         return isSequenceFound;
50     }
51
52     void receiveData(std::vector<uint8_t>& byteVector) {
53         std::lock_guard<std::mutex> lock(queueMutex);
54         if (byteDataQueue.size() < 100) {
55             byteDataQueue.push(byteVector);
56         }
57     }
58

```

```

59     void transitionToNextModule(std::shared_ptr<IModule>
nextModule) {
60         std::lock_guard<std::mutex> lock(queueMutex);
61         this->registerProcessingCallback([this, nextModule](
std::vector<uint8_t>& byteVector) {
62             nextModule->receiveData(byteVector);
63         });
64     }
65
66     void registerProcessingCallback(ProcessingCallback
callback) {
67         if (callback) {
68             callbackFunction_ = callback;
69         } else {
70             std::cerr << "[SequenceSearcher] Invalid callback
." << std::endl;
71         }
72     }
73
74     void processQueueDataAndInvokeCallback() {
75         while (isRunning) {
76             std::unique_lock<std::mutex> lock(queueMutex);
77             if (!byteDataQueue.empty()) {
78                 std::vector<uint8_t> byteVector =
byteDataQueue.front();
79                 byteDataQueue.pop();
80                 lock.unlock();
81                 bool isMatchFound = processVector(byteVector)
;
82                 if (isMatchFound) {
83                     std::cout << "[SequenceSearcher] Match
found, forwarding..." << std::endl;
84                     if (callbackFunction_) {
85                         try {
86                             callbackFunction_(byteVector);
87                         } catch (const std::exception& e) {
88                             std::cerr << "[SequenceSearcher]
Callback error: " << e.what() << std::endl;
89                         }
90                     } else {
91                         std::cerr << "[SequenceSearcher] No
callback set!" << std::endl;
92                     }
93                 } else {

```

```

94         std::cout << "[SequenceSearcher] No
matching data found." << std::endl;
95     }
96     } else {
97         lock.unlock();
98         std::this_thread::sleep_for(std::chrono::
milliseconds(50));
99     }
100 }
101 }
102
103 private:
104     std::thread processingThread;
105     bool isRunning = false;
106     std::mutex queueMutex;
107     std::condition_variable logDataAvailable;
108     std::queue<std::vector<uint8_t>> byteDataQueue;
109     ProcessingCallback callbackFunction_;
110 };

```

Questions

1. Class Design and Purpose

- What is the purpose of the `SequenceSearcher` class? Explain its main functionality.
- Why is the constructor initialized with `isRunning` set to `false`?

2. Thread Management

- How does the `start()` method work? What happens when it is called?
- What is the role of the `isRunning` variable in the `processQueueDataAndInvokeCallback` method?
- Explain the purpose of the `stop()` method. What happens if the thread is not joinable?
- Why is `std::thread` used, and why is a lambda function passed as an argument to it?

3. Data Processing

- (a) How does the `processVector()` method work? What is the purpose of the `TARGET_SEQUENCE`?
- (b) What is the role of `std::search` in the `processVector()` method?
- (c) What happens if the target sequence is found in the byte vector?

4. Mutex Usage

- (a) Why is `queueMutex` used in the `SequenceSearcher` class? What problem does it solve?
- (b) What happens if the `queueMutex` is not used in the `receiveData()` method?
- (c) Why is `std::unique_lock` used in the `processQueueDataAndInvokeCallback()` method instead of `std::lock_guard`?

5. Callback Mechanism

- (a) What is the purpose of the `registerProcessingCallback` method? How is it used in the code?
- (b) In the `transitionToNextModule` method, why is the callback registered with a lambda function? What does this lambda function do?
- (c) How does the callback mechanism work in the `processQueueDataAndInvokeCallback()` method?

6. Error Handling

- (a) How does the code handle errors in the callback function? What happens if an exception is thrown?
- (b) What happens if the `callbackFunction_` is not set when `processQueueDataAndInvokeC` is called?

7. Code Improvements

- (a) Are there any potential issues with the current implementation of `processQueueDataAndInvokeCallback()`? How would you improve it?
- (b) How would you modify the code to allow for configurable sleep durations between data processing?

Scoring

Each question is worth 5 points. The total score is out of 40 points.

Good Luck!