[Квалификационный тест](https://lib.sd.avia-cosmos.ru/pages/viewpage.action?pageId=61505643) C++(Qt)

**Тестовое задание:**

Разработка клиент-серверного приложения на Qt C++ для телекоммуникационной системы

**Цель:**

Проверить навыки кандидата в разработке сетевых приложений с использованием Qt C++, включая работу с сетевыми сокетами (QTcpServer, QTcpSocket), обработку JSON-данных, создание GUI с использованием QMainWindow и QTableWidget, а также эмуляцию данных.

**Задание**

Необходимо разработать клиент-серверное приложение для системы. Сервер представляет собой GUI-приложение, которое управляет подключенными клиентами (устройствами) и отображает их данные в таблицах. Клиент — консольное приложение, эмулирующее устройство, которое передает серверу данные различного типа и длины (метрики сети, статус устройства, логи).

Задачи

1. Сервер (GUI-приложение:

- Разработать приложение с графическим интерфейсом на базе QWidgets (если используется QML — огромный плюс, но не обязательно)

- Сервер должен:

- Слушать входящие подключения на порту 12345.

- Поддерживать одновременное подключение нескольких клиентов.

- Отображать список подключенных клиентов в таблице (QTableWidget) с колонками: ID клиента, IP-адрес, статус (подключен/отключен).

- Отображать в отдельной таблице получаемые от клиентов данные в формате: ID клиента, тип данных (например, "NetworkMetrics" (необходимо распарсить по ключам для отображения в таблицу), "DeviceStatus"(тоже необходимо распарсить), "Log"), содержимое данных, время получения.

- Предоставлять возможность конфигурировать параметры клиентов через настройки (например критические значения метрик или статуса, при которых начнётся отправка отдельных Log-пакетов с предупреждением)

- Обрабатывать JSON-данные от клиентов и отправлять подтверждение подключения клиента в формате JSON.

- Интерфейс приложения должен включать:

- Кнопку для запуска/остановки работы клиентов.

- Две таблицы для отображения информации о клиентах (текущие подключения, например ip, порт) и данных, отправляемых клиентами.

- Лог событий (например, QTextEdit) для вывода сообщений о подключении/отключении клиентов и ошибках.

2. Клиент (консольное приложение):

- Разработать консольное приложение, эмулирующее устройство.

- Клиент должен:

- После запуска пытаться подключиться к серверу на localhost:12345. Если подключение не удалось, повторять попытки каждые 5 секунд.

- После успешного подключения ожидать подтверждение подключения, потом ожидать команду на старт, затем начинать отправку JSON-данных трех типов:

- NetworkMetrics: метрики сети (например, {"type": "NetworkMetrics", "bandwidth": 100.5, "latency": 12.3, "packet\_loss": 0.01}).

- DeviceStatus: статус устройства (например, {"type": "DeviceStatus", "uptime": 3600, "cpu\_usage": 25, "memory\_usage": 60}).

- Log: текстовый лог (например, {"type": "Log", "message": "Interface eth0 restarted", "severity": "INFO"}).

- Данные отправляются с случайной задержкой от 0.01 до 0.1 секунд.

- Длина содержимого варьируется: короткие сообщения (до 50 символов), средние (50–200 символов), длинные (200+ символов, например, для логов).

- Обрабатывать подтверждения от сервера на подключение и выводить их в консоль.

- При отключении от сервера пытаться переподключиться.

Требования к реализации

- Использовать Qt 6.5.2

- Для сетевого взаимодействия применять QTcpServer и QTcpSocket.

- Логика сервера вынесена в отдельный поток от его GUI

- Данные передаются в формате JSON с использованием QJsonDocument и QJsonObject.

- Сервер должен корректно обрабатывать одновременное подключение нескольких клиентов.

- Клиент должен генерировать данные с использованием случайных значений (например, с помощью QrandomGenerator).

- Код должен быть чистым, с комментариями и разделением логики на классы/функции.

- Обработать возможные ошибки (например, некорректный JSON, потеря соединения).

Критерии оценки

1. Корректность работы:

- Сервер запускается, принимает подключения, отображает данные и позволяет управлять клиентами.

- Клиент подключается, отправляет данные разных типов и обрабатывает ответы.

2. Качество кода:

- Читаемость (использовать Google c++ GuideLines), модульность (разделение на логические классы), наличие комментариев

- Корректное использование сигналов и слотов Qt.

3. GUI сервера:

- Наличие приемлемого интерфейса, корректное обновление таблиц, корректная обработка логики кнопок (старт, стоп и т.д).

- Реализация функционала для настройки клиентов.

4. Реализованный функционал:

- Количество (и качество) реализованного функционала в API клиент-сервера (сам функционал не определён, можно придумать самому, но в пределах тематики «управляющее GUI-приложение и подключаемые устройства»

5. Обработка ошибок:

- Устойчивость к некорректным данным и сбоям соединения.

6. Наличие готовых инструкций сборки в проекте

Пример сценария работы

1. Запускается сервер, отображается GUI с пустыми таблицами.

2. Запускается несколько клиентов, каждый подключается к серверу.

3. Сервер отправляет пакеты с подтверждением подключения клиентов

4. В таблице клиентов сервера появляются записи с ID, IP и статусом.

5. Клиенты отправляют данные (например, NetworkMetrics, DeviceStatus, Log).

6. Сервер отображает данные в таблице данных

7. Лог сервера отображает события: подключение, получение данных, отключение.

**Ожидаемый результат**

- Исходный код двух приложений (сервер и клиент) с инструкцией по запуску.

- CMakeLists или .pro файлы для сборки проектов в Qt Creator.

- Краткое описание логики работы (в README или комментариях).

- Будет плюсом: папка build, в которой находится exe-файл с решением, который запускается без QtCreator.

Рекомендации

- Для GUI сервера используйте QMainWindow, QTableWidget, QPushButton, QTextEdit.

- Для генерации случайных данных в клиенте применяйте QRandomGenerator и QTimer.

- Для обработки нескольких клиентов на сервере храните сокеты в QMap или QList.

- Используйте сигналы и слоты для асинхронной обработки событий.

Рудаков Александр