S02.E04.Build2023.03.29.0

Задача 2. JSON

0. Материалы

Nlohmann:

А) Исходные тексты для работы с JSON в C++

Б) Автоматизированная установка для Debian/Ubuntu: `sudo apt-get install nlohmann-json-dev`

В) Автоматизированная установка для MS Visual Studio

HTTPLib:

А) Исходные тексты для работы с HTTPLib в C++

CMake:

А) Установщик для Windows

Б) CMake Reference Documentation

В) CMake Tutorial

Д) Практические примеры работы с CMake [Habr]

1. Общие требования

А) Задача должна быть оформлена в виде класса с реализацией соответствующих функций;

Б) Реализация интерфейса должна содержать вызов соответствующих функций с запросом у пользователя необходимых параметров. Интерфейс реализации может быть разработан в одном из двух типов:

I. Использование «мнемонических цифр», определяющих вызов соответствующих функций с предложением дополнительного ввода запрашиваемых параметров;

II. Разбор (парсинг) введенных данных по формату, определяющих вызов соответствующих функций с дополнительными параметрами;

2. Описание частей задачи.

Часть А: Ввод и вывод данных

- в функции *Input()* произвести считывание файла, содержащего одну или несколько структурных текстовых данных в формате JSON, в форматированную JSON-структуру библиотеки nlohmann;
- в функции *Print()* реализовать консольный вывод (вывод на экран) из текущей форматированной JSON-структуры библиотеки nlohmann.

Часть Б: Обработка данных с использованием JSON

- реализация функции условия задачи в функции *Execute()*;

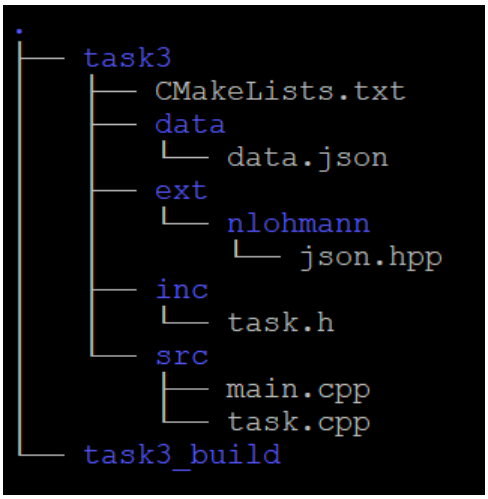
- реализация функции добавления единичных данных *Add()* в форматированную JSON-структуру библиотеки nlohmann (данные могут быть добавлены по определенному наименованию поля, однозначно определяющего добавляемую структуру, остальные данные могут быть выбраны случайным образом по предопределенному набору данных);
- реализация функции удаления данных *Delete()* из форматированной JSON-структуры библиотеки nlohmann (удаление заданных данных по определенному полю, однозначно определяющего удаляемую структуру);
- реализация функции генерирования данных *Generate()* для формирования заданного пользователем числа форматированных данных JSON-структуры (для строковых полей значение может выбираться из заранее заданного списка случайным образом, остальные числовые значения также могут быть заполнены случайным образом).

Часть В: Файл сборки CMakeLists для Части А и Б

Реализация должна быть оформлена в виде:

- файла описания класса (в виде заголовочного файла *.h) в папке `inc`;
- файла описания функций класса (в виде файла с исходным кодом *.cpp) в папке `src`;
- дополнительных вспомогательных средств (например, nlohmann) в папке `ext`;
- (по желанию) подготовленных входных данных в папке `data`;

Примерное расположение файлов в директории:



- Формирование проекта должно производиться во внешней директории с постфиксом «`_build`» (например, `task3_build`) с генератором по умолчанию командой: `cmake ../task3`
- Сборка выходных исполняемых файлов из сформированного проекта должна осуществляется во внешней директории с постфиксом «`_build`» (например, `task3_build`) командой `cmake --build .`

Часть Г: Обработка данных с использованием HTTPLib

Реализация, произведенная в части А и Б должна быть разделена на 2 изолированные части:

I) Часть I (клиент) реализует и оперирует командами для работы с со структурой данных (форматированными JSON-данными библиотеки nlohmann), хранящимися в части II (сервер). В интерфейсе части I (клиент) должны быть реализованы команды отсылки/получения сообщения и завершения работы.

Отправляемые сообщения имеют вид:

Структура данных *Send*:

строка Command (наименование или мнемоническое обозначение команды),

массив строк Param (второстепенные параметры)

Получаемые сообщения имеют вид:

Структура данных *Receive*:

строка Command (наименование или мнемоническое обозначение команды),

целое число Result (контрольное значение для проверки, 1 – операция выполнена успешно, 0 – операция прервана).

По результату полученного ответного сообщения вывести об успешности реализованной команды.

II) Части II (сервер) является хранителем структуры данных (форматированных JSON-данных библиотеки nlohmann). При получении очередной команды от Части I (клиент) исполняет ее и демонстрирует результат работы. Для данной части не подразумевается пользовательского интерфейсного ввода.

Дополнительно (не обязательно): класс для обработки команд Части II (сервер) может быть реализован в виде прикрепляемой динамической библиотеки к основной программе с экспортируемыми функциями.

Подразумевается, и Часть I (клиент), и Часть II (сервер) могут отсылать и принимать сообщения с использованием собственного (выделенного) или общего порта. В рамках задачи предполагается, что взаимодействие происходит с использованием локальной машины 127.0.0.1. Рекомендуемый порт для обмена - 8080, дополнительный порт для обмена - 8081.

Часть Д: Формирование дополнительного поля Идентификатор объекта

Для каждой структуры данных сформировать уникальный идентификатор определенного формата (формат может быть произвольным, однако должен включать постоянную часть).

S02.E04.Build2023.03.29.0

Пример формата:

`<Наименование структуры данных><Порядковый номер (3 цифры)><Наименование(первые 3 буквы)>`

Интерфейс программы подразумевает проверку запрашиваемого идентификатора по шаблону с использованием регулярных выражений и выдачу ассоциированных данных.

Часть Е: Описание исходного кода программы

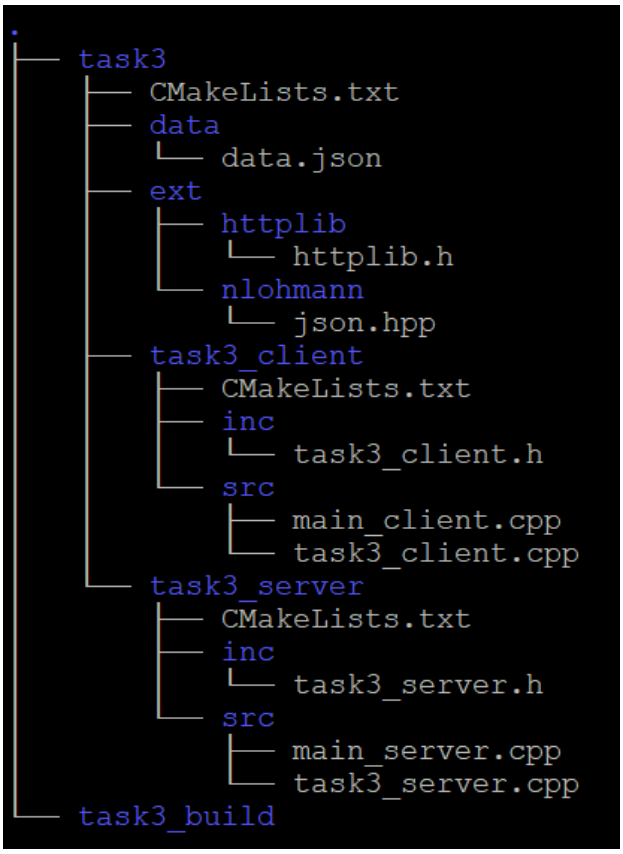
С использованием инструментария Doxygen подготовить в заголовочных файлах клиентской и серверной части описание назначения класса, назначение используемых функций класса (в том числе принимающих аргументов функций и возвращаемого значения.

Выходной обработанный заголовочный файл может быть оформлен в виде формата HTML или PDF.

Часть Ж: Файл сборки CMakeLists для Части Г и Д

Адаптированный вариант сборки Части В с учетом изменения структуры программы в Части Г и Д для формирования двух исполняемых файлов.

Примерное расположение файлов в директории:



S02.E04.Build2023.03.29.0

1. Структура данных *Literature*:

массив строк *Author* (список авторов),

строка *Name* (наименование издания),

целое число *Year* (год издания),

строка *Publisher* (наименование издательства),

вещественное число *Rating* (рейтинг издания).

Функция:

Вывести отсортированные данные в файл в формате JSON, оставив 10% книжной продукции с максимальным и минимальным рейтингом (поле *Rating*).

2. Структура данных *Mountain*:

строка *Name* (наименование издания),

строка *Country* (страна),

вещественное число *Altitude* (высота),

вещественное число *Latitude* (широта),

вещественное число *Longitude* (долгота).

Функция:

Вывести отсортированные данные в файл в формате JSON по стране (поле *Country*) и по высоте (поле *Altitude*).

3. Структура данных *City*:

строка *Name* (наименование),

строка *Country* (страна),

целое число *Population* (популяций),

массив *Rating* (рейтинг), включающий строку *Name* (наименование рейтинга), вещественное число *Value* (значение рейтинга).

Функция:

Вывести в файл в формате JSON все записи с большим числом популяции (поле *Population*) и с тем же именем страны (поле *Country*), где занесен максимальный рейтинг (*Rating*) из всех записей с именем страны (поле *Country*).

4. Структура данных *Country*:

строка *Name* (наименование),

целое число *Population* (популяций),

вещественное число *Square* (площадь),

массив *Rating* (рейтинг), включающий строку *Name* (наименование рейтинга), вещественное число *Value* (значение рейтинга)

Функция:

Вывести в файл в формате JSON записи за исключением записей, содержащие наименьшую площадь (поле *Square*), рейтинг (поле *Rating*) и популяцию (поле *Population*).

5. Структура данных *Train*:

целое число *Index* (номер вагона),

вещественное число *Total* (вместимость (т.)),

массив *Freight* (груз), включающий строку *Type* (тип груза) и вещественное число *Weight* (вес груза (т.)).

Функция:

Вывести в файл в формате JSON записи за исключением записей, содержащие вместимость (поле *Total*) меньше заданной пользователем величины.

6. Структура данных *Good*:

строка *Stor* (наименование магазина),

строка *Name* (наименование товара),

вещественное число *Price* (цена товара),

вещественное число *Weight* (вес товара),

целое число *Total* (наличие, число шт.).

Функция:

Вывести в файл в формате JSON все записи, в которых перераспределено число товаров (поле *Total*) таким образом, чтобы в разных магазинах (поле *Stor*) оказалось одинаковое число товаров с указанным наименованием (поле *Name*).

7. Структура данных *Event*:

строка *Name* (наименование),

строка *City* (город),

строка *Date* (время начала),

целое число *Population* (численность).

Функция:

Вывести в файл в формате JSON все записи, в которых перераспределены даты событий (поле *Date*) таким образом, чтобы каждый город (поле *City*) имел крупное событие по численности (поле *Population*).

8. Структура данных *Restaurant*:

строка *Name* (наименование),

вещественное число *Bill* (средний чек),

целое число *Menu* (число вариантов меню),

массив *Rating* (рейтинг), включающий строку *Name* (наименование рейтинга), вещественное число *Value* (значение рейтинга).

Функция:

Вывести отсортированные данные в файл в формате JSON в зависимости от числа вариантов меню (поле *Menu*) и заданного рейтинга (поле *Rating*).

9. Структура данных *Film*:

строка *Country* (страна),

массив строк *Actor* (актеры),

строка *Name* (наименование),

целое число *Year* (год издания).

Функция:

Вывести в файл в формате JSON записи за исключением записей с ранним годом выпуска (поле *Year*).

10. Структура данных *Route*:

строка *Start* (начальный пункт),

строка *End* (окончательный пункт),

вещественное число *Length* (длина маршрута (км.)),

массив строк *Transport* (тип транспорта).

Функция:

Вывести в файл в формате JSON записи, соответствующие определенному типу транспорта (поле *Transport*), заданного в исходных данных.

11. Структура данных *CookingRecipe*:

строка *Name* (наименование),

массив *Ingredient*, включающий строку *Name* (наименование), вещественное число *Value* (вес ингредиента в граммах),

вещественное число *Time* (длительность приготовления),

вещественное число *Cal* (калорийность).

Функция:

Вывести в файл в формате JSON записи в зависимости от заданного диапазона калорийности (поле *Cal*) и диапазона времени приготовления (поле *Time*).

12. Структура данных *Coin*:

строка *Country* (страна),

строка *Currency* (валюта),

целое число *Nominal* (номинал),

вещественное число *Mass* (масса).

Функция:

Вывести в файл в формате JSON записи с наименьшим номиналом (поле *Nominal*) и массой монеты (поле *Mass*).

13. Структура данных *Metro*:

строка *City* (город),

строка *Name* (наименование станции),

целое число *Line* (номер линии),

вещественное число *Capacity* (пропускная способность тыс.чел/час).

Функция:

Вывести в файл в формате JSON записи, содержащие наименование города (поле *City*), число линий в виде целого числа (поле *Lines)* и число станций метро (поле *Stations*).

14. Структура данных *Company*:

строка *City* (город),

строка *Name* (наименование компании),

массив строк *Assignment* (назначение),

вещественное число *Finance* (оборот).

Функция:

Вывести в файл в формате JSON записи с запрошенным у пользователя диапазоном по финансовому обороту (поле *Finance*).

15. Структура данных *Site*:

строка *Name* (наименование интернет-ресурса),

вещественное число *Audience* (оборот),

массив строк *Topic* (тематика),

массив *Rating* (рейтинг), включающий строку *Name* (наименование рейтинга), вещественное число *Value* (значение рейтинга)

Функция:

Вывести отсортированные данные в файл в формате JSON по аудитории (поле *Audience*) и числу тематик (поле *Topic*).

16. Структура данных *Aircraft*:

строка *Name* (наименование производителя),

строка *Model* (наименование модели),

целое число *Distance* (дистанция),

вещественное число *Capacity* (вместимость).

Функция:

Вывести в файл в формате JSON отсортированные записи по наименованию модели (поле *Model*), содержащие наименование производителя (поле *Name*), наименование модели (поле *Model*) и соотношение полей вместимость (поле *Capacity*) и дистанция (поле *Distance*) в виде вещественного числа (поле *Efficiency*).

17. Структура данных *Star*:

строка *Name* (наименование),

строка *Date* (дата открытия),

массив *Sputnik* (спутник), включающий строку *Name* (наименование), вещественное число *Mass* (масса)

вещественное число *Mass* (масса),

вещественное число *RotationPeriod* (период вращения (часы)).

Функция:

Вывести в файл в формате JSON отсортированные записи по числу спутников (поле *Sputnik*).

18. Структура данных *Weather*:

вещественное число *Temp* (температура),

структура *Wind* (ветер), включающая строку *Direction* (направление), вещественное число *Speed* (сила ветра, м/с)

вещественное число *Humidity* (влажность),

вещественное число *Pressure* (значение давления).

Функция:

Вывести в файл в формате JSON отсортированные записи по температуре (поле *Temp*), оставив только положительные значения.