План реализации проекта.

СОДЕРЖАНИЕ

1. ***Краткое описание задачи.***

Необходимо сделать скрипт для поиска и замены содержания в .json файлах. Дополнительно было решено написать скрипт для генерации этих самых файлов по шаблону, а также разбить функцию поиска на два вида: замена по всем файлам .json файла и замена в определенном файле .json файла, то есть в .json файле находится массив с данными вида {name : “” , content : “”}, и под name подразумеваются названия файлов с содержанием, далее именуемые, как ***файлсодержание,*** которые могут иметь расширение .csv (таблица) && .txt (текст), и замена во всех файлах .json файла подразумевает замену всех заданных подстрок в каждом файлесодержании, а замена в определенном фале .json файла подразумевает замену всех заданных подстрок в определенном файлесодержании.

2. ***Цель и ожидаемый результат.***

Цель в большей мере прописана в задаче, а конкретно создать скрипт для поиска и замены содержания в файлахсодержании с режимом “ТОЛЬКО ПОИСК” и итоговым отчетом о сесии. Дополнительно создать скрипт, создающий .json файлы по шаблону {name : “” , content : “”}.

Ожидается следующий функционал:
На запуске генерируется рандомное количество .json файлов в количестве от 5 до 10 шт., далее эти .json файлы заполняются рандомным количеством строк, в формате {name : “” , content : “”} в установленном диапазоне (400к-500к) в равной степени каждый, то есть этим диапазоном устанавливается общее количество строк, которое в равной степени разделяется между всеми сгенерированными .json файлами, кроме последнего, в который уходит помимо основного количества еще и неделимый остаток, то есть пример: Указывается диапазон 60-100 строк, после генерируется из рандомного числа условно 5 .json файлов.

Далее генератор выбирает из диапазона 60-100 сгенерировать 76 строк.
76 делится на 5 файлов, значит в каждом будет по 15 файлов, кроме последнего, в котором будет 16 строк.
Далее эти объекты вида {name : “” , content : “”} заполняются типизированным именем **file_номер файла.расширение** и контентом, который зависит от расширения файла: если это .csv файл, то он заполняется случайным количеством столбцов в заданном диапазоне, и в каждом столбце заполняется случайным количеством случайных латинских строчных букв, количество столбцов и букв в содержании генерируется рандомно в заданном диапазоне, что позволяет генерировать практически уникальное содержание в каждом файлесодержании, а если это .txt файл, то генерируется случайное количество случайных строчных латинских букв в заданном диапазоне, так же для уникализации содержания.
Далее пользователю сообщается об успешной генерации .json файлов и предлагается ввести команду для дальнейшей работы, если пользователь не знает команд, то ему также предлагается ввести команду **--help**, которая выводит список доступных для ввода команд, а именно:

- Получение информации о доступных командах - "--help"
- Поиск слова в содержании файлов - "--find"
- Замена содержимого во всех файлах - "--replaceall"
- Замена содержимого только в одном файле - "--replacein"
- Режим "ТОЛЬКО ПОИСК" - "--dry-run"
- Завершение работы и удаление json файлов - "--exit"

В случае ввода команды **--find** пользователю выдается сообщение с просьбой ввода искомой подстроки, после которого происходит поиск заданной подстроки по всем строкам всех .json файлов. Далее выводится результат поиска в следующем виде:

| название .json файла | название файласодержания | содержание строки |

и в конце выводится общее количество найденных совпадений.

В случае ввода команды **--replaceall**, которая отвечает за замену всех подходящих подстрок в выбранном .json файле, пользователю выводится сообщение с просьбой ввести данные в следующем виде:

название .json файла без расширения -> что заменить -> на что заменить

после чего происходит замена заданной подстроки на новую, с выводом пользователю сообщения о количестве совершенных замен, а в случае безуспешной замены, то есть отсутствия заданной подстроки в файлахсодержании, выводится сообщение о нулевом количестве замен. В случае ввода неверного названия .json файла выводится сообщение о том, что данный файл не был найден.

В случае ввода команды **--replacein**, которая отвечает за замену всех подходящих подстрок в выбранном .json файле и выбранном пользователем файлесодержании, пользователю выводится сообщение с просьбой ввести данные в следующем виде:

название .json файла без расширения -> название файла содержании -> что заменить -> на что заменить

после чего происходит замена заданной подстроки на новую, с выводом пользователю сообщения о количестве совершенных замен в содержании заданного файласодержания, а в случае безуспешной замены, то есть отсутствия заданной подстроки в заданном файлесодержании, или отсутствия заданного файласодержания в указанном .json файле, выводится сообщение о нулевом количестве замен. В случае ввода неверного названия .json файла выводится сообщение о том, что данный файл не был найден.

В случае ввода команды **--exit** будет совершено удаление всех сгенерированных .json файлов с последующим завершением работы программы и выводом пользователю сообщения об этом.

В случае ввода команды **--dry-run** будет включен режим “ТОЛЬКО ПОИСК”, который подразумевает использование только режима поиск, то есть команду **--find**, в данном режиме не доступны режимы замены, соответственно недоступны команды **--replacein** и **--replaceall**, но остаются доступными команды **--help** *&&* **--exit**, а также появляется команда **--dry-end**, которая завершает режим “ТОЛЬКО ПОИСК”, и переводит работу программы в обычный режим.

В случае ввода неизвестной команды выведется сообщение об ошибке, что команда неизвестна, для получения списка доступных команд будет предложено ввести команду **--help**.

Также, после завершения работы программы, формируется отчет, в котором находятся следующие данные:

- Дата и время отчёта.
- Продолжительность сессии.
- Общее количество введённых команд.
- Общее количество найденных совпадений.
- Общее количество выполненных замен.
- Количество замен по файлам со списком файлов с заменами.

3. ***Этапы и задачи реализации.***

Анализ задачи -> Продумывание генератора .json файлов -> Продумывание функции поиска -> Продумывание функции замены по всем файламсодержаниям -> Продумывание функции замены в определенном файлесодержании -> Продумывание функции для удаления .json файлов и формирования отчета -> Продумывание дружелюбного интерфейса -> Тестирование+исправления -> Оформление

4. ***Сроки по этапам.***

25.11.2025 – вторник – открылся дедлайн, изучил проект, написал план реализации необходимых функций, начал писать генератор.

26.11.2025 – среда – продолжаю писать генератор, понял, что генерация .json файлов со стоками на тысячи знаков в контенте будет очень долгой, даже после оптимизации путем удаления генерации отступа у объектов, а также замены классической конкатенации на потоковую запись, генерация происходила ощутимо долго, также было тяжело работать с выводом таких огромных строк, тк вывод даже одной строки на 1000+ символов был проблематичный, а при нахождении тысяч совпадений при поиске, на вывод бы шли миллионы символов, и форматировать такой вывод в обычной консоли было бы невозможно, поэтому поменял генерацию на менее объемную, максимум 30 символов в одном содержании файласодержания.

28.11.2025 - 30.11.2025 – пятница – воскресенье – допилил генерацию и работал над основным функционалом проекта, а конкретно поиск, замена двух видов, режим «ТОЛЬКО ПОИСК».

01.12.2025 – понедельник – принес проект на тестовый показ, узнал что нужно сделать, уточнил непонятные моменты, по приезду домой начал допиливать уже созданные функции, исправил выводы в консоль, тк они были ломаные.

05.12.2025 – 07.12.2025 – пятница – воскресенье – кинул генератор в .h заголовочный файл, написал небольшой бенчмарк для измерения скорости генерации и поиска, создал функцию, которая завершает работу программы с удалением созданных .json файлов, для того, чтобы каждый перезапуск программы имел полный смысл, добавлена возможность завершения работы без удаления .json файлов, а также запуска без генерации, то есть работы с уже имеющимися .json файлами, допилил оформление всех выводов и тп, добавлен итоговый отчет после заверения работы программы о проделанной работе. Сделано полное оформление проекта для сдачи.

08.12.2025 – понедельник – первая полноценная попытка сдачи итоговой лабораторной работы.

5. ***Ресурсы.***

Chatgpt.com – для более удобного поиска информации.

https://stackoverflow.com/questions/12233710/how-do-i-use-the-ostringstream-properly-in-c - информация про потоковую запись ostringstream

https://stackoverflow.com/questions/36751133/proper-method-of-using-stdchrono - информация о std::chrono

https://stackoverflow.com/questions/65169630/visual-studio-code-using-filesystem-library - информация о std::filesystem