**ПРОЕКТ**

ТЕМА: „**JSON Парсер**“

Автор:

Ванина Тенева, Курс: 1, Група: 2

СЪДЪРЖАНИЕ

[1 Увод 3](#_Toc199913128)

[2 Цели и обхват на софтуерното приложение 4](#_Toc199913129)

[3 Анализ на решението 4](#_Toc199913130)

[4 Дизайн 5](#_Toc199913131)

[5 Ефективност и бързодействие на решението 5](#_Toc199913132)

[6 Тестване 6](#_Toc199913133)

[7 Заключение и възможно бъдещо развитие 6](#_Toc199913134)

# Увод

Текущото приложение представлява конзолен JSON парсър и редактор, разработен на езика C++ с цел демонстрация на работа със структурирани данни в текстов формат, без използването на външни библиотеки. Основната му функционалност включва зареждане, парсване, визуализиране, търсене, редактиране и управление на JSON документи, следвайки строго зададения синтаксис на JSON формата.

Програмата предоставя възможност за:

* Зареждане на JSON съдържание и изграждане на вътрешно представяне чрез йерархична структура от обекти
* Навигация и манипулация по път чрез команди
* Търсене по израз на произволно ниво и извеждане на намерените стойности.
* Запазване на редактирани JSON документи във файла
* Валидация на входните данни и подходящо извеждане на съобщения при грешки.

Основният акцент в проекта е поставен върху обектно-ориентирания дизайн. Използвайки абстрактен клас, реализираме наследяването на конкретни JSON типове. Осигурена е гъвкавост на всички стандартни JSON структури.

# Цели и обхват на софтуерното приложение

Целта на приложението е да предостави функционален JSON парсър, изцяло реализиран на C++ без външни библиотеки, който позволява анализ, модификация и управление на JSON документи.

Приложението покрива основни операции по обработка на данни и цели да даде практическа представа за работа със структури от данни.

# Анализ на решението

Решението е изградeно на базата на **обектно-ориентиран подход**, като JSON структурата се представя чрез абстрактен клас JsonNode и неговите производни JsonObject, JsonArray и JsonValue. Парсването на входен JSON текст се реализира ръчно чрез рекурсивни методи, които изграждат съответната дървовидна структура в паметта.

Програмата поддържа работа с пътища до елементи, което позволява директна навигация и манипулация на вложени стойности. Конзолната апликация покрива всички изисквани операции: print, search, set, create, delete, move, saveas.

Използван е минимален брой външни зависимости, като това демонстрира добро владеене на основните структури и механизми в C++..

# Дизайн

Приложението е структуирано с помощта на модулна обектно-ориентирана архитектура, която отразява структурата на JSON формата.

Парсването се реализира в класа JsonParser, който ръчно анализира входния текст и изгражда дървовидна структура. Основната логика за търсене, създаване, редактиране, изтриване и записване на елементи е капсулирана в съответните класове, с ясни методи като getByPath, setByPath, createByPath, deleteByPath, и search.

Конзолната апликация използва разбиване на път чрез вектори и работи директно върху парснатото дърво, като по този начин позволява динамично управление на данните.

# Ефективност и бързодействие на решението

Приложението работи с йерархична структура от обекти, представена чрез указатели и стандартни контейнери (std::vector, std::shared\_ptr), което осигурява добра производителност при обхождане, търсене и актуализация на JSON структурата. Операциите по достъп и манипулация на данни чрез пътища се реализират рекурсивно и са ефективни за типични по размер JSON документи.

# Тестване

Приложението е тествано с разнообразни JSON структури, включително вложени обекти и масиви. Проверена е коректността на основните операции: парсване, pretty print, търсене по ключ, създаване, редактиране и изтриване. При тестването са включени и гранични случаи като невалиден вход, непълен път и опити за промяна на несъществуващи елементи, като програмата реагира с подходящи съобщения за грешка.

# Заключение и възможно бъдещо развитие

Създаденото приложение успешно изпълнява основните изисквания на заданието за JSON парсър и редактор. То поддържа парсване на валидни JSON структури, търсене на ключове, създаване, промяна, изтриване, както и записване на резултата във файл. Решението е обектно-ориентирано, модулно и лесно за разширяване.

Възможно бъдещо развитие включва поддръжка на допълнителни JSON типове (числа, булеви стойности, null), по-добра обработка на грешки, графичен потребителски интерфейс, както и възможност за валидиране на JSON схемата.