JSON Parse

от Цветомир Стайков

1MI0800469 2 група 1 курс

1. Увод

1.1 Описание и идея на проекта

- Проекта представлява програма в стил конзолен интерфейс (*Command line interface*), в която потребителя може да зарежда *JSON* (*Javascript Object Notation*) файлове. Тя дава възможност за манипулиране, извеждане и проверка на данните от файловете. Написана е в обектно ориентиран стил и по начин, чрез който части от програмта може да се имплементират в други проекти.

1.2 Цел и задача на разработка

- Проекта е направен да дава възможност за манипулирането, проверяване и извеждане на *JSON* файлове. *И*ма команди за проверка на валидността на заредения файл, която проверява дали спазва *JSON* формата, да извежда информацията от файла, във четим формат (*„pritty print“*), да модифицира информацията и да запазва както в досегашния файл, така й на нови места. Програмата е написана и тествана на *Linux Fedora*. Обектно ориентирания стил на програмата и начина, по който е написана, дава възможност да се разширява програмта и да се имплементира в други проекти.

1.3 Структура на документация

- 1. Увод
    1.1. Описание и идея на проекта

    1.2. Цел и задачи на разработката

    1.3. Структура на документацията

- 2. Преглед на предметната област

    2.1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани

    2.2. Дефиниране на проблеми и сложност на поставената задача

    2.3. Подходи, методи (евентуално модели и стандарти) за решаване на поставените проблемите

    2.4 Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...) и качествени (нефункционални) изисквания (скалируемост, поддръжка, ...)

- 3. Проектиране

    3.1. Обща архитектура – ООП дизайн

3.2. Диаграми (на структура и поведение - по обекти, слоеве с най-важните извадки от кода)

- 4. Реализация, тестване

4.1. Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода)

4.2. Управление на паметта и алгоритми. Оптимизации.

4.3. Планиране, описание и създаване на тестови сценарии (създаване на примери)

- 5. Заключение

5.1. Обобщение на изпълнението на началните цели

5.2. Насоки за бъдещо развитие и усъвършенстване

2. Преглед на предметната област

2.1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани

- *JSON* или *JavaScript Object Notation,* е популярен формат за съхранение на данни в дървеловиден вид. Често използван за изпращане на информация през мрежата, съхранение на конфигуриращи файлове (*configs*) и много други. Лесно може да се разчете без специален софтуер, но има стриктна структура и правила, за който най-добре да се използва помощ. Добре описани са в сайта **json.org**

2.2. Дефиниране на проблеми и сложност на поставената задача

- Проблема, който трябва да се реши, включва взимане на даден файл, анализиране с цел разделя на данните и запазването в някаква стуктура, чрез която да се позволи лесната обработка и извеждане на данните. Структурата на входния файл, ако е валиден, е лесна за разбиране, но се оказа голямо предизвикателство за имплементиране. Имплементирането и тестването на системата се оказа значително по-трудно от първоначално предвидената замисъл.

2.3. Подходи, методи (евентуално модели и стандарти) за решаване на поставените проблемите.

- За работа с проекта беше избрана с команден интерфейс (command line interface). За решението на дадения проблем, беше създаден главен клас *JsonParser,* и няколко отделни калса за съхранение на информацията. Беше създаден функция валидатор, който да минава през данните и да ги съхранява и допълнителни функци за работа с файловата система.

2.4 Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...) и качествени (нефункционални) изисквания (скалируемост, поддръжка, ...)

- За работа с програмта не е нужно администраторски достъп или някакви допълнителни специални права, но програмта трябва да има достъп до чете и писане на компютъра. Всички команди могат да се извикват следкато програмата е стартирата. По-голямата част от кода може да се използва и в друг и програми, но някой функционалности (като някои извеждането на грешки) са директно вградени в главните класове и трябва да се модифицират от там.

3. Проектиране

3.1. Обща архитектура – ООП дизайн

- Има няколко главни класа – „Rooter“, „Objects“, „Values“, като те са наравени така че да може да се извеждат подкласовете чрез абстракция. „Objects“ има два подкласа „jsonObject“ и „jsonArray“, а за „Values“ са изведени всички възможни стойности, който „jsonObject“ и „jsonArray“ може да съхраняват. Това са „vNumber“, „vString“, „vBoolean“, „vArray“, „vObject“. Класовете „Objects“ и „Values“, имат различни абстрактни функци, което принуди класовете да са разделни. Същия проблем имат „jsonObject“ и „jsonArray“.

Понеже класа „Values“ е се използва да се напише една обща функция „ void* getData(unsigned int) “, която да бъде прехвърлена на подкласовете му. Идеята е да се връща „празен“ показател (void pointer), който обаче да сочи към някаква информация, след което според вида клас, който връща информацията, този показател да се трансформира в нужната стойност.

3.2. Диаграми (на структура и поведение - по обекти, слоеве с най-важните извадки от кода)

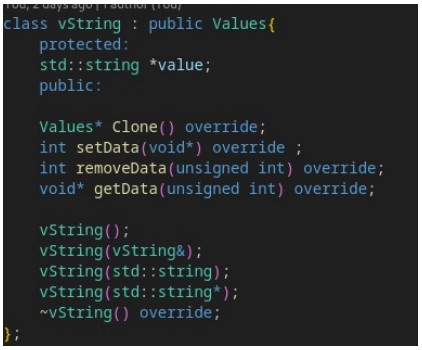
4. Реализация, тестване

4.1. Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода)

-Всеки клас има динамична памет, като това представлява показател към стойност, за класовете от тип „Values“, а при класовете от тип „Objects“ това е показател от показатели, като идеята е да се инициализира масив от показатели към други обекти. При това беше създадена функция клониране, която да връща копие на сегашния елемент. Бяха създадени виртуални деструктори за ичистване на динамичната памет.

```
You, 2 days ago | 1 author (You)
class jsonObject : public Objects{
    protected:
    Pair** pairs;

    void expand() override;
    void shrink() override;

    public:
    jsonObject();
    jsonObject(jsonObject&);
    Objects* Clone() override ;

    void AddPair(Pair * pair);
    void AddPair(std::string key, Values* value);

    Pair* ReturnPair(std::string key);
    Pair* ReturnPair(unsigned int index);

    int RemovePair(std::string key);
    int RemovePair(unsigned int index);
    ~jsonObject() override;
};
```

```
You, 2 days ago | 1 author (You)
class Objects : public Rooter{

    protected:
    size_t size;
    virtual void expand() = 0;
    virtual void shrink() = 0;
    public:
    size_t Size();
    virtual Objects* Clone() = 0;
    virtual ~Objects() {}
};
```



4.2. Управление на паметта и алгоритми. Оптимизации.

- Алгоритъма за обработка на данните е сръвнително дълъг и беше пренаписан няколко пъти, за да покрива различни невалидни стойности и за по-лесна работа с него при нуждата от подобрения. Главната му идея е че минава през данните като има няколко ключови елемнта като скоба за нов елемент или масив („ {, }, [,] „), двоеточие за намиране на следващ елемент („ : „) , или запетая на нов елемент („ , „). При намиране на скоба трябва да се извика функцията рекурсивно от момента до където е стигнала сега и да се продължи обработката на данните там, като това което тази функция върне ще е нов елемент в досегашния ключ. Когато функцията се върне тя ще изгнорира следващите елементи, когато не намери затварящата скоба на сегашната отворена и така ще продължи докато не стигне до края на функцията. Скоростa не мога да изчисля, понеже той е рекурсивен и зависи от елементите в данните, но смятам, че най-лошия вариант е O(n + k*m), където n е входните данни, k размера на най-големия елемент (започвайки от отворена скоба до затварящата я) и m – колко подобни елемнти се срещат.

4.3. Планиране, описание и създаване на тестови сценарии (създаване на примери)

-Тестовете, които бяха направени имаха за цел да тестват дали обектите от различните класове правилно съхраняват информация, дали могат да я обработват и да я връщат.

- Тук беше намерен проблем, при които в някои тестове при изтриване на показатели, други странични тестове биват повлиявани. При премахването на показателите или на целия тест, другите тестове тръгваха.

5. Заключение

За жалост