

Проект „Json-Parser“
по
Обектно-Ориентирано
Програмиране

Изготвил:Йордан Павлов FN№71978

Съдържание:

Глава 1. Увод

- 1.1. Описание и идея на проекта
- 1.2. Цел и задачи на разработката
- 1.3. Структура на документацията

Глава 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, ...)

Глава 3. Проектиране

- 3.1. Обща архитектура – ООП дизайн
- 3.2. Диаграми (на структура и поведение - по обекти, слоеве с най-важните извадки от кода)

Глава 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)

Глава 5. Заключение

- 5.1. Обобщение на изпълнението на началните цели
- 5.2. Насоки за бъдещо развитие и усъвършенстване

Използвана литература

Глава 1.Увод

- 1.1 Проектът има за цел да редактира файлове в формат json.Тя трябва да има потребителски интерфейс ,така че чрез команди да може лесно да се работи със нея .
- 1.2 Основната цел на програмата е да се използва лесно чрез команди поради тази причина е създаден клас Json който съдържа необходимите команди,а те са именно създаване на дума по зададена друга, премахване на дума, преместване на две думи, заместване да дума с друга,запазване на дума в произволен файл и в файл зададен от потребителят и при стартиране на програмата тя автоматично проверява файла дали е в формата и ,ако има грешки показва думата или символът след .Също файла може да се изпρινтира в конзолата.
- 1.3 Структурата на документацията се извършва по следните стъпки.Първата е да се представяне на належащите проблеми.Втората е търсенето на достатъчно ефективно решение.Документацията завършва с представянето на различни примери и сценарии с цел да се онагледят поведението на програмата в различните сценарии.

Глава 2.Преглед на предметна област

2.1 За целта на проекта ще бъде използван обектно-ориентирана на програмирането.Тя ще позволи поддръжката на програмата да бъде достатъчно лесна и опростена.

2.2 Трябва да бъдат разрешени няколко проблема за реализацията на пълният проект.Основният е свързан с функционалността трябва да се проверяват данните от потребителя за коректност и по-конкретно дали дадените думи от него се съдържат в файла. Друг е свързан с потребителския интерфейс като трябва да се следи за безпроблемната работа с програмата.

2.3 Подходът към решаването на вече зададените проблеми е да създам необходимите методи с помощта на който да коректно изпълнение на програмта и извеждане коректни данни на потребителя.

2.4 Има няколко потребителски изисквания за проекта.Те са свързани с лесната и правилна употреба.Функционални изисквания са правилната работа с файла, както и правилното заделяне на динамична памет където е необходимо. Програмата не трябва да натоварва устройството на потребителя с излишни ресурси.Поддръжката трябва да бъде лесна, защото при възникнали проблеми да бъдат лесно открити и отстранени.

Глава 3.Проектиране

3.1 Обектно-ориентираният дизайн се състои в реализацията на клас Json_parser,Command и Json. Json_parser съдържа член данните ключ който е променлива от тип стринг и стойност който е променлива от тип стринг.

Json съдържа съответно методите за добавяне на дума, размятане на две думи, замятане на дума, изтриване на дума, принтиране на файла в конзолата и валидиране на файла, запазване на дума в произволен или зададен от потребителя файл .

Класът Command съдържа методи за отваряне и зареждане на данните в програмата , затваряне, запазване на файл, както и помощ с необходимите команди за работа с файла.

Глава 4.Реализация, тестване

4.1. Реализация на класове

```
class Json_struct{
private:
    string key;
    string value;
}

void validate(const char* file_path);
void print(const char* file_path);
void search(const string &key);
void set(const string &path, const string &word_for_replace);
void create(const string &path, const string &word_for_create);
void deletefunc(const string &path);
void move(const string &from_path, const string &to_path);
void save(const string &path);
void save_as(const string &file_path, const string &path);
void read_from_file(const string &file_path);
void print_vector();
void save_to_file(const string &file_path);
```

4.2. използвал съм вектори, които сами си освобождават паметта.

4.3. validate experiment.json

```
open
replace "New_York", "sofia",
delete "home",
create "street": "my_town_is",
move "firstName": "lastName":
print expo.json
save_file "bulgaria"
save_as_file expo.json "sofia"
```

Глава 5.Заклучение

5.1. Проектът остава да се допълни със търсене по заден път от потребителят. Бяха намерени оптимални решения за решаването на проблемите които бяха зададени.Интерфейсът е достатъчно функционален и лесен за употреба.

5.2. Могат да се добавят още функции като например за създаване на деца по зададен родител.

Използвана литература

<http://www.cplusplus.com/> -за функцията find

<https://www.geeksforgeeks.org/> - за сплитването на командтата по думи