**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ВТ**

**ОТЧЕТ**

**по Курсовой работе №1**

**по дисциплине «Программирование»**

**Тема:**

**ИЗУЧЕНИЕ И ПРАКТИЧЕСКОЕ ОСВОЕНИЕ ПРИЕМОВ ПРОГРАММИРОВАНИЯ ФУНКЦИЙ ДЛЯ ПРЕДСТАВЛЕНИЯ И ВЫПОЛНЕНИЯ ЭЛЕМЕНТАРНЫХ ОПЕРАЦИЙ С ТЕКСТОМ И ЛИНЕЙНЫМ СПИСКОМ.**

|  |  |  |
| --- | --- | --- |
| **Студент гр. 9005** |  | **Самуйлов. Е.С.** |
| **Преподаватель** |  | **Калмычков В.А.** |

**Санкт-Петербург**

**2020**

**Цель работы.**

Изучение и практическое освоение приемов программирования функций для представления и выполнения элементарных операций с текстом (символьной информацией) из строк с маркером на основе структур (классов) с использованием массива (обычного/динамического) и организации работы с файлами (чтение/запись текста) с использованием библиотек stdio или fstream, оформления фрагментов программы (типов и функций) в виде многомодульной реализации (набор файлов в проекте).

**Основные теоретические положения.**

Основные теоретические положения. Для выполнения было получено задание на разработку программы. В этой программе необходимо использовать линейный односвязный список для хранения информации. Общим функционалом является работа с файлами и вывод полученных результатов в файл.

**Задание**

Текст представляет собой последовательность отдельных предложений, содержащих слова и знаки пунктуации.

Преобразовать текст в соответствии с последовательностью команд редактирования,

которые должны позволять вставлять, удалять и заменять заданные слова в определенных предложениях.

Команды редактирования:

Р.2) вставить в предложении новое слово перед заданным словом,

Р.6) удалить в предложении знак пунктуации (указанный и/или все).

Указание определенного предложения:

H.2) предложение, начинающееся с указанного слова,

Указание заданного слова:

С.7) содержащее заданную последовательность символов

**Внешний формат хранения данных**

Для работы используется 2 файла

1. для взятия данных
2. для вывода результатов работы

**Внутренний формат хранения данных**

Данные одного предложения хранятся в классе и берутся по необходимости   
class Sentence

{

public:

char \*\*sentenceWith = new char \*[CHAR\_MAX];

int sWCounter = 0;

char \*\*punctuation = new char \*[CHAR\_MAX];

int \*wordLength = new int[INT16\_MAX];

int nwLength = 0;

int pCounter = 0;

int startOnThisWordNumber = 0;

Все вычитанные данные хранятся в sentenceWith, для которого в sWCounter хранится информация о количестве слов в предложении. Информация о длине каждого отдельного слова хранится в wordLength.

Данные, которые имеют отношение ко всему тексту хранятся в структуре myStore

struct MyStor {

string nameOfFile; - имя файла вывода

int sentenceCounter = 0; - порядковый номер предложения

string commandLine;- строка команд

int pos = 0;- позиция для Р 2

int blockerP2 = 0;- блокировщик для метода P2

int blockerH2 = 0;- - блокировщик для метода H2

int blockerC7 = 0;-- блокировщик для метода C7

int blockerP6 = 0;-- блокировщик для метода P6

char \*newWord = new char[CHAR\_MAX];- новое слово для метода Р2

char \*word = new char[CHAR\_MAX];- слово для метода С7

int wLength = 0;- длина слова из метода С7

};

**Методы, классы, структуры**

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя структуры\класса\метода** | **Назначение** | **Параметры** | **Возвращаемое значение** |
| **main** | **точка старта программы** | **-** | **0** |
| **класс Sentence** | **Обработка одного предложения по заданным параметрам и вывод его в указанный файл** | **Введенное предложение** | **-** |
| **Структура myStore** | **хранение всей информации, необходимой для экземпляров класса Sentence** | **nameOfFile - имя файла вывода**  **commandLine - строка команд** | **-** |
| **метод splitter** | **разбирает предложение на части и вызывает необходимые методы** | **string line - предложение** | **-** |
| **метод add** | **создает новое слово и отделяет от него знаки препинания при наличии** | **char \*wordAdd - указатель на добавляемое слово int length - длина добавляемого слова** | **-** |
| **метод punctuationChecker** | **возвращает информацию о том, является ли присланный символ знаком пунктуации** | **char mark** | **bool**  **thrue - если является и false если нет** |
| **метод commandParser()** | **берет строку команд из структуры myStore и разбирает на команды, по которым вызывает методы** | **-** | **-** |
| **метод deletePunctuationMark** | **удаление всех или конкретных знаков пунктуации** | **int isDeleteAll - флаг удаления всех знаков char \*punctMarc указатель на удаляемый знак int length длина удаляемого символа (если многоточие)** | **-** |
| **метод startOnThisWord** | **проверяет, является ли данное слово первым в обрабатываемом предложении** | **char \*word - указатель на слово  int length - длина слова** | **-** |
| **метод**  **usingSumbolSequence** | **проверяет предложение на наличие слова, содержащего последовательность wordUse** | **char \*wordUse - указатель на последовательность**  **int length - длина последовательности** | **-** |

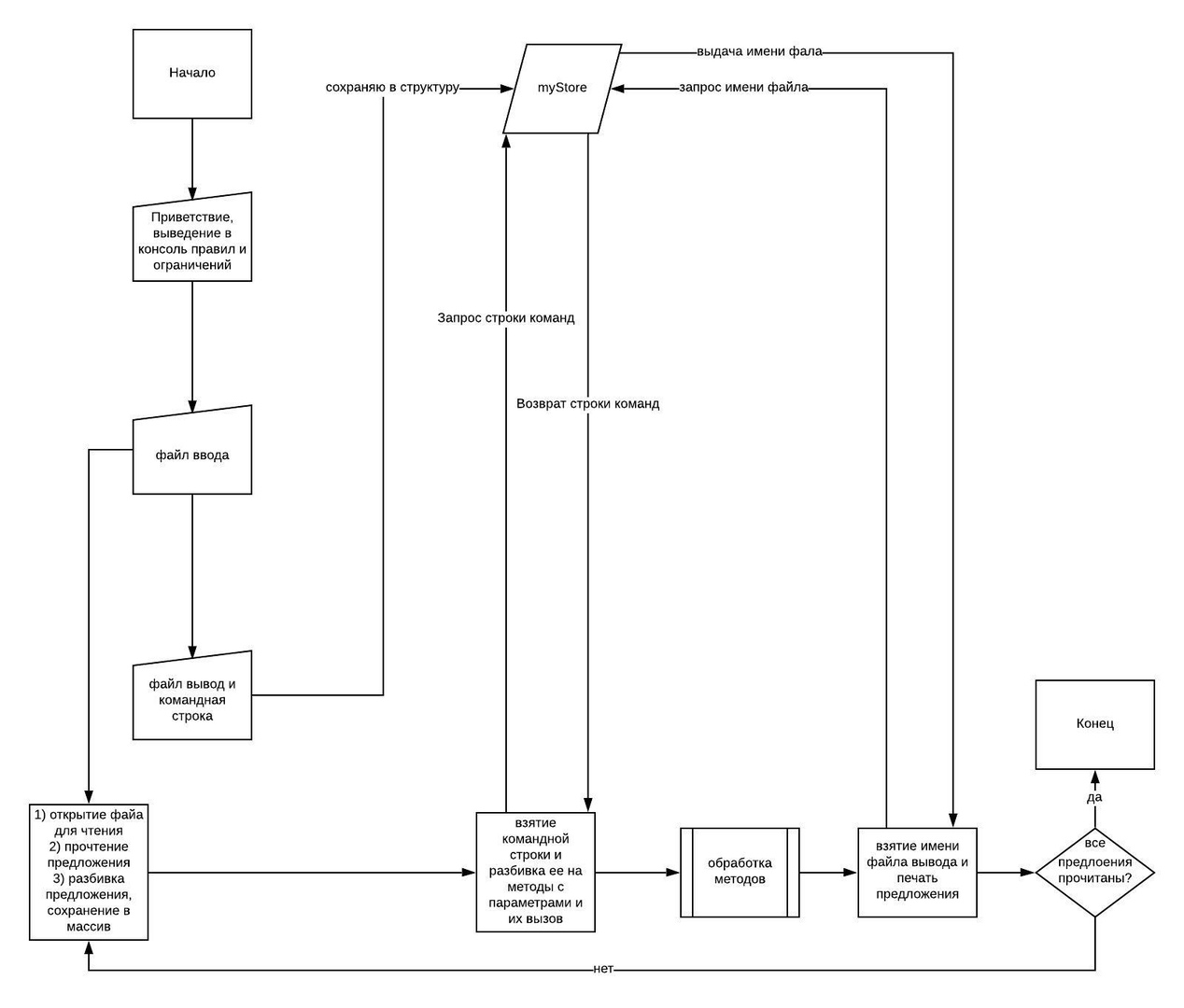
|  |  |  |  |
| --- | --- | --- | --- |
| **метод printer()** | **Выводит предложение** | **-** | **-** |

**Алгоритм обработки данных**

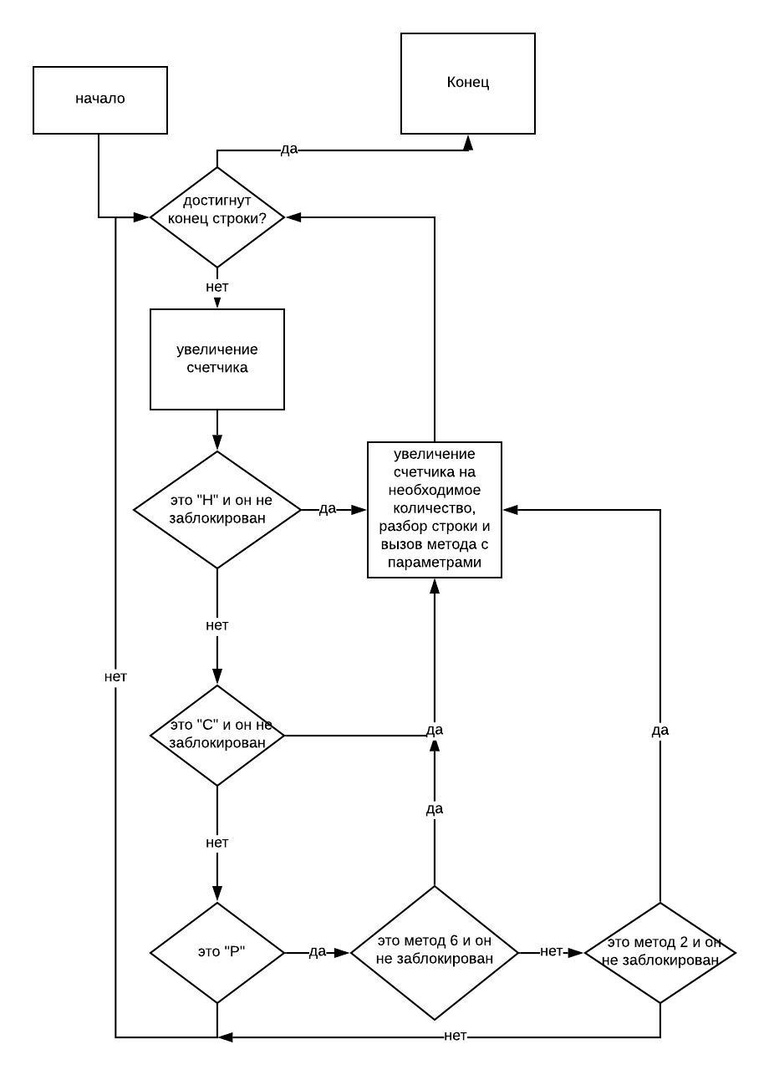
Программа выводит приветствие и описание всех функций, возможностей и правил ввода

затем при получении имя файла вывода и строка команд передаются в структуру, а файл ввода открывается для чтения. Для каждой строки создается новый экземпляр класса, в который передается полученная строка, внутри класса строка разбивается на слова, а затем выделяются и знаки пунктуации, после чего вызывается метод для разбивки строки команд, которые сразу же вызываются, после чего по завершении этих процедур вызывается метод для печати, который дозаписывает информацию в файл.

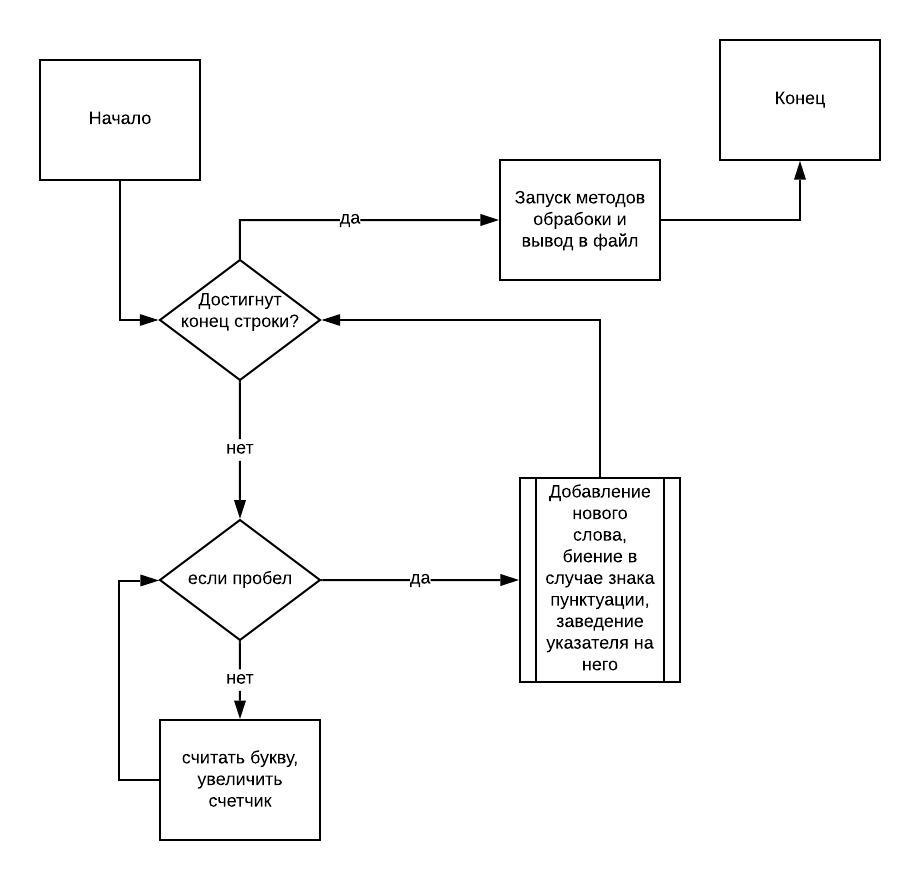
**Блок-схема(общая)**

****

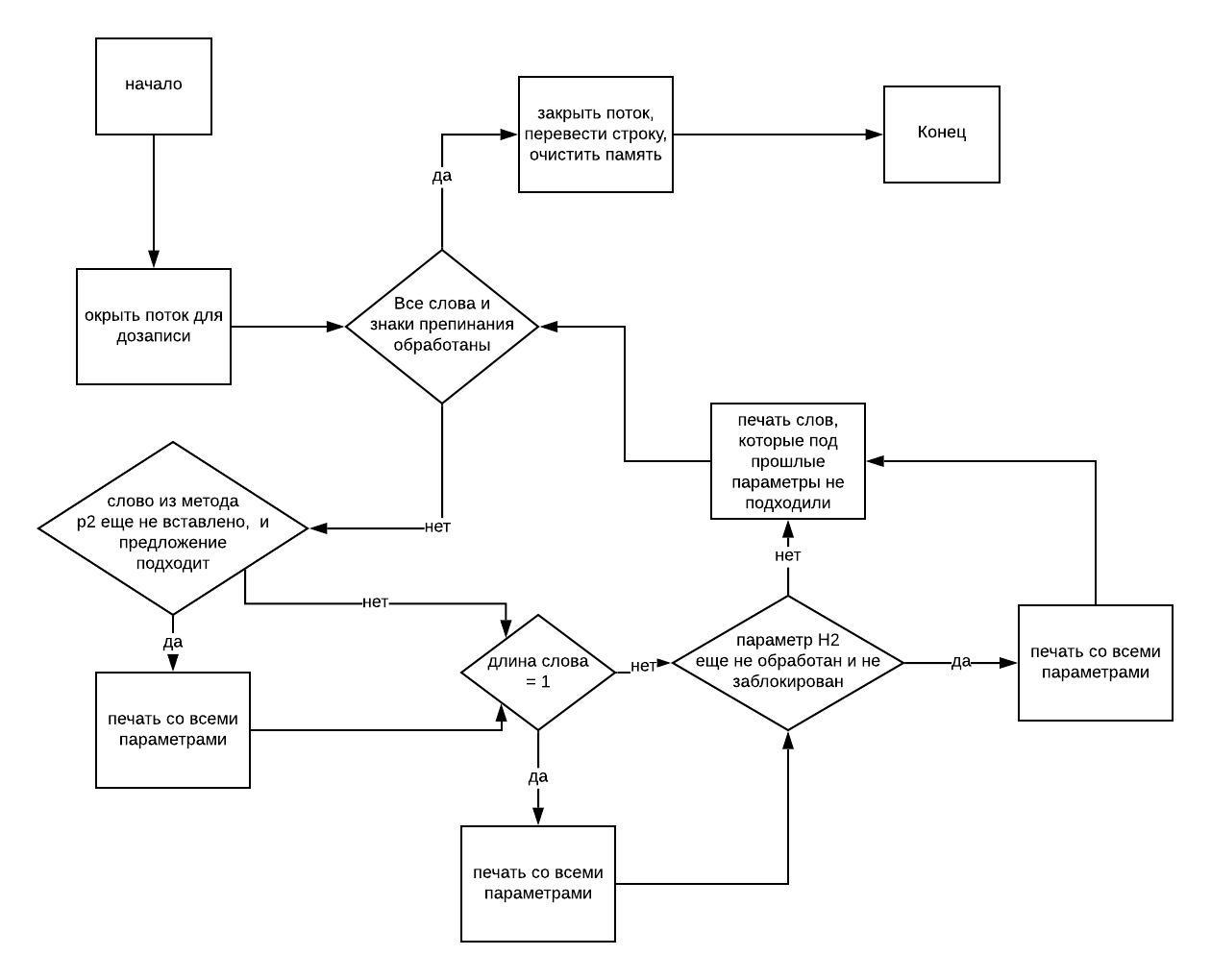
**Блок-схема (command parser)**

****

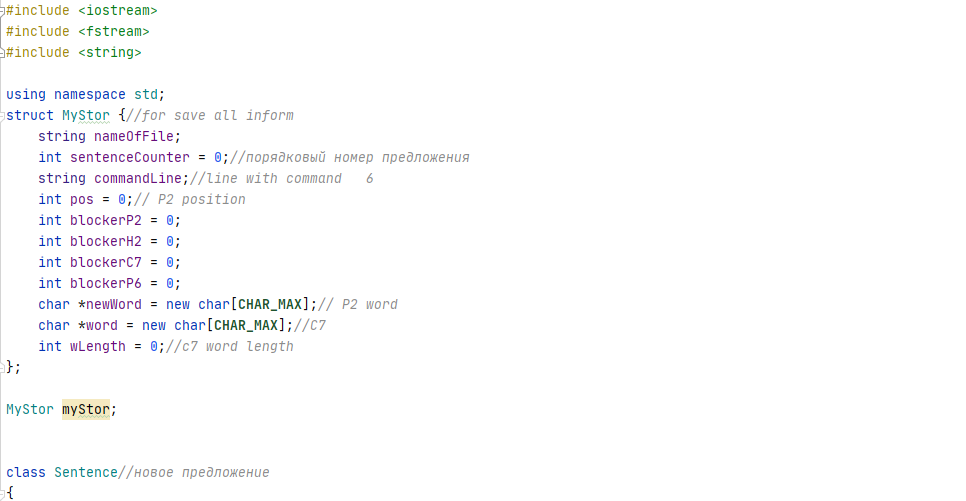
**Блок-схема(метод add, метод split)**

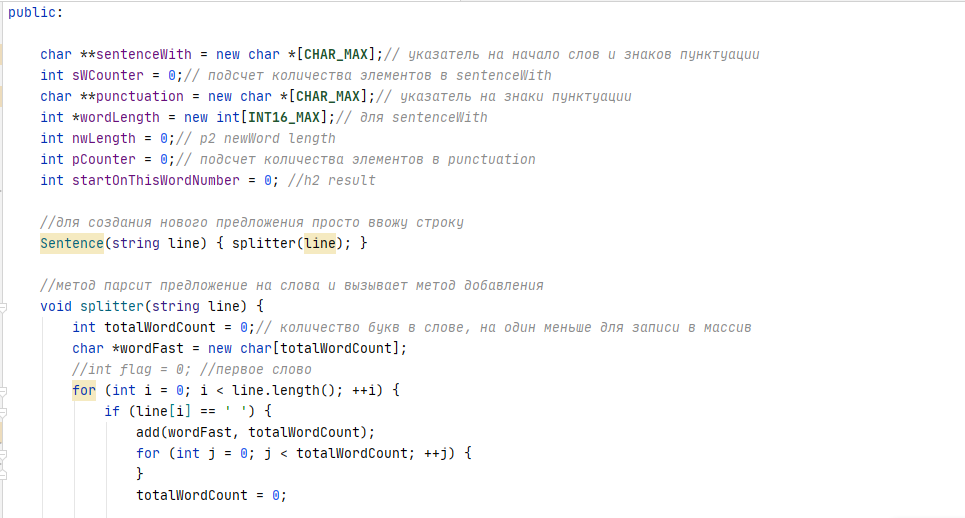
****

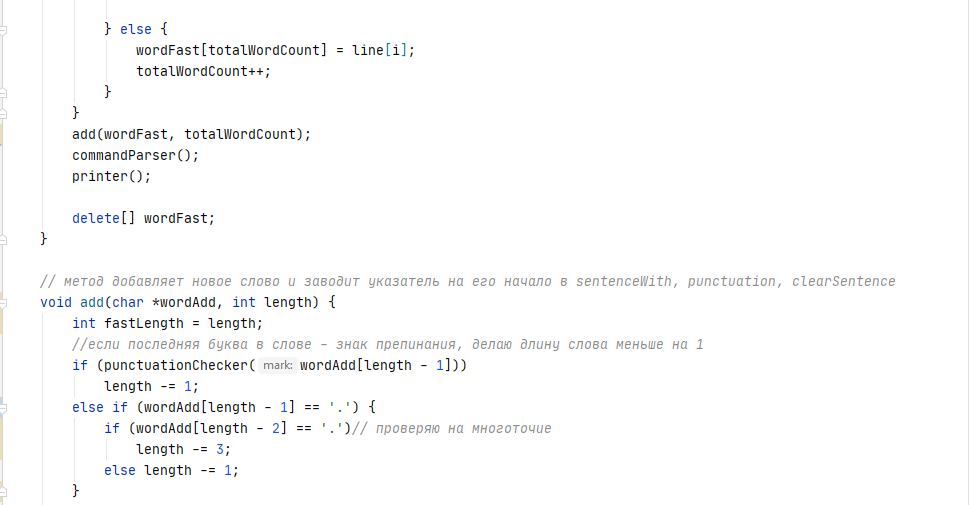
**Блок-схема(метод printer)**

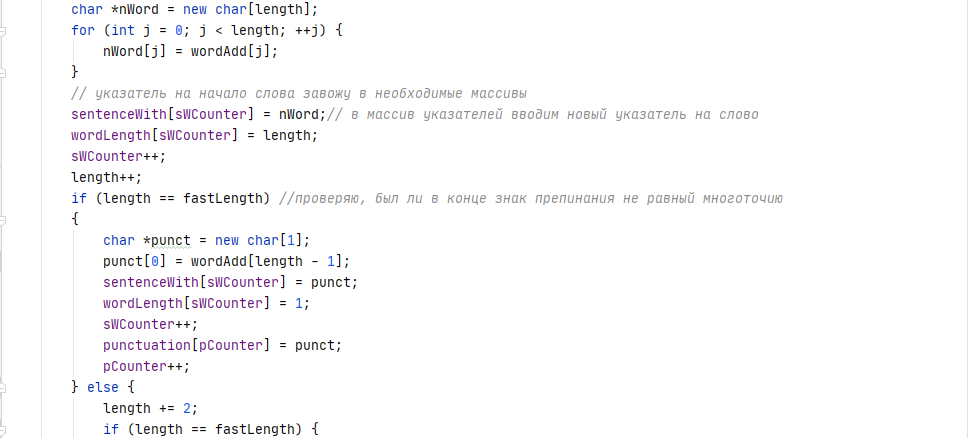
****

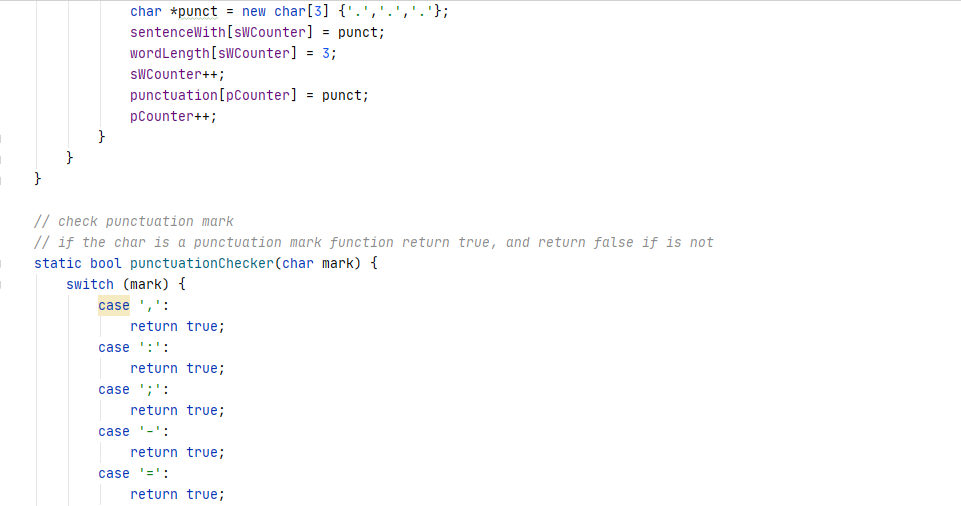
**Код программы**

****

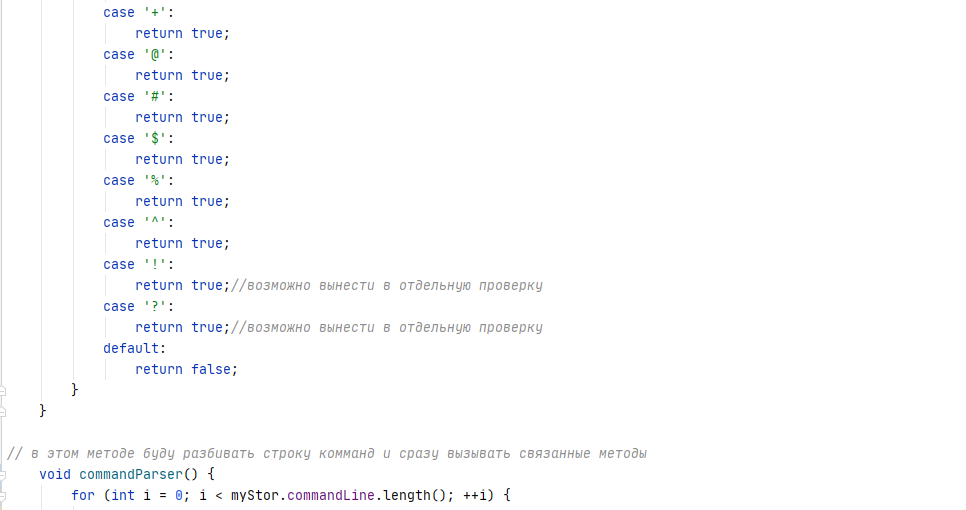
****

****

****

****

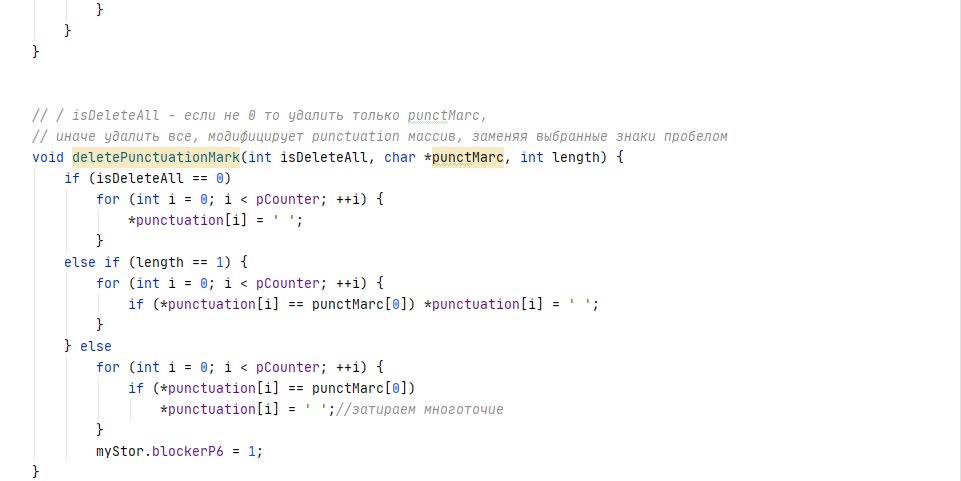
****

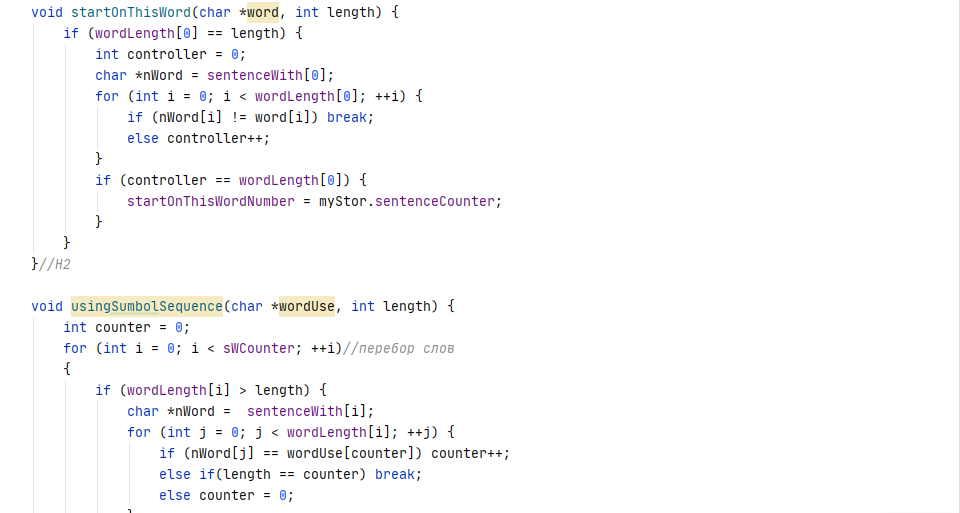
****

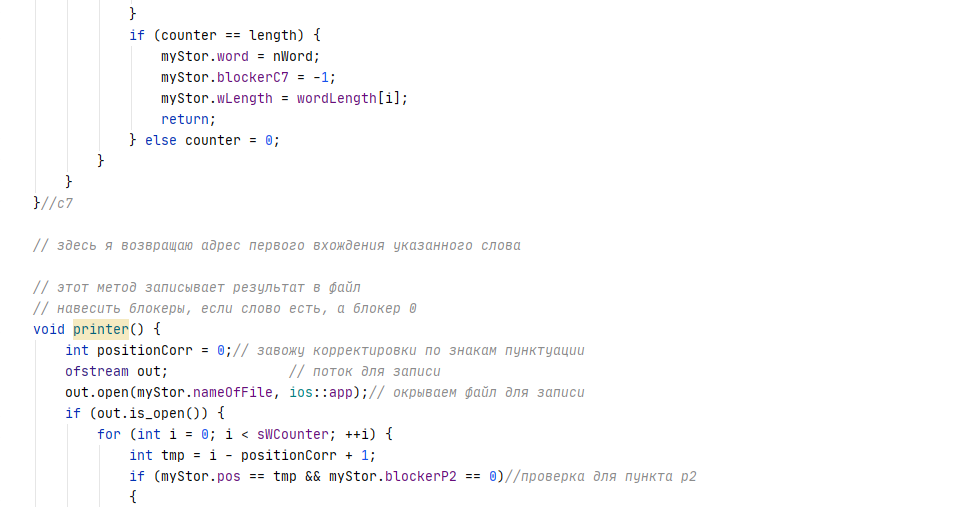
****

****

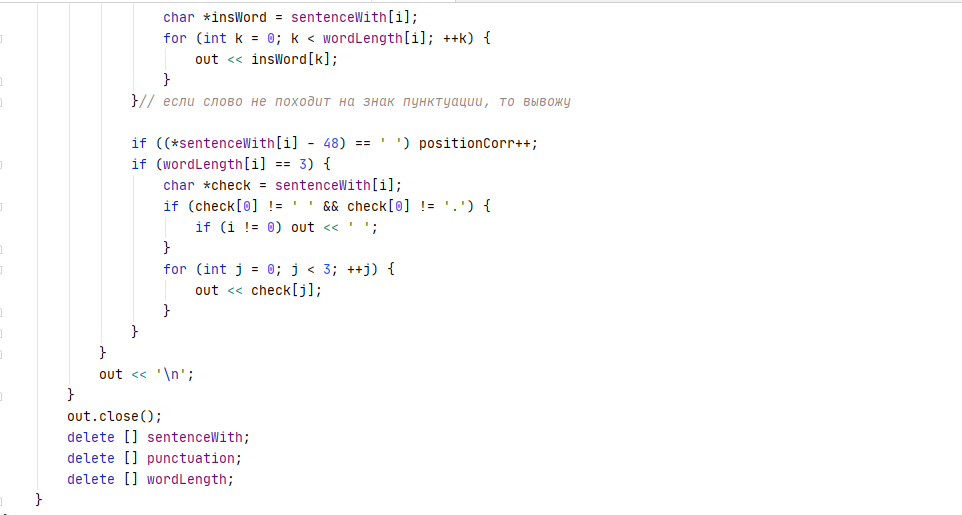
****

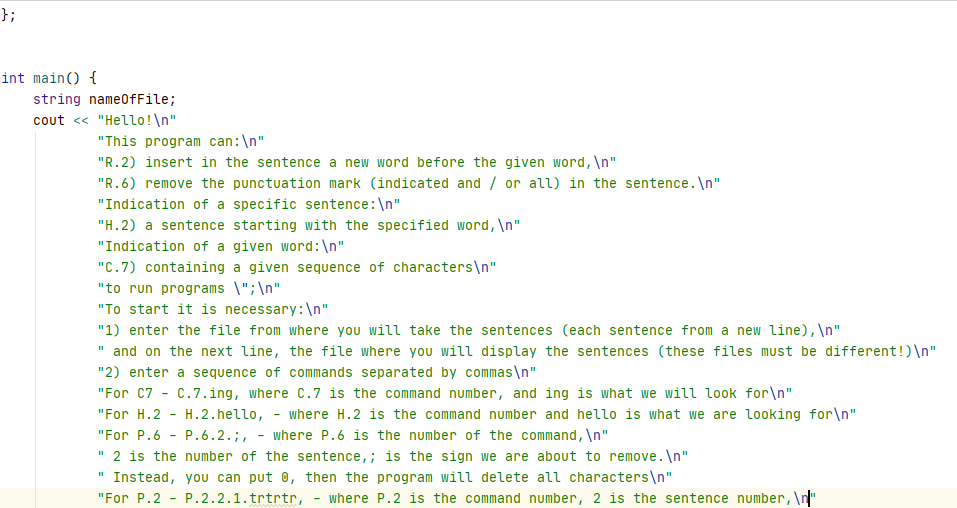
****

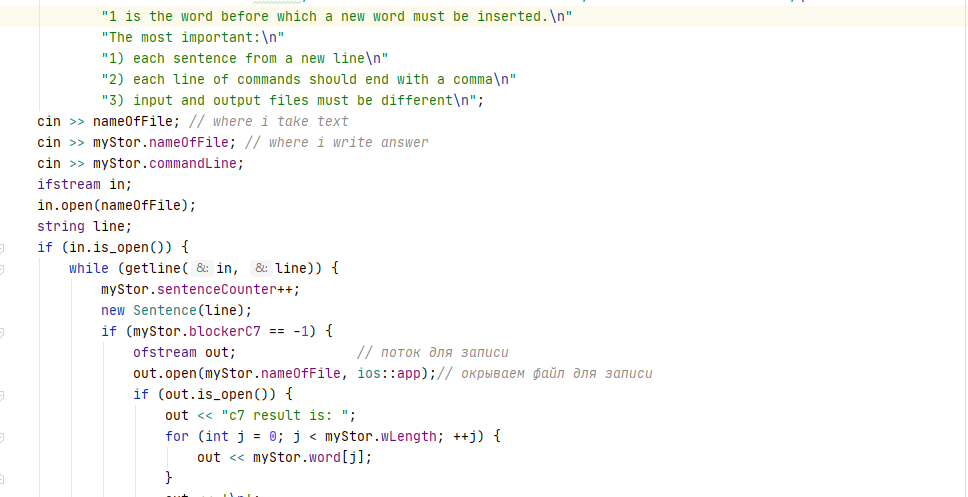
****

****

****

****

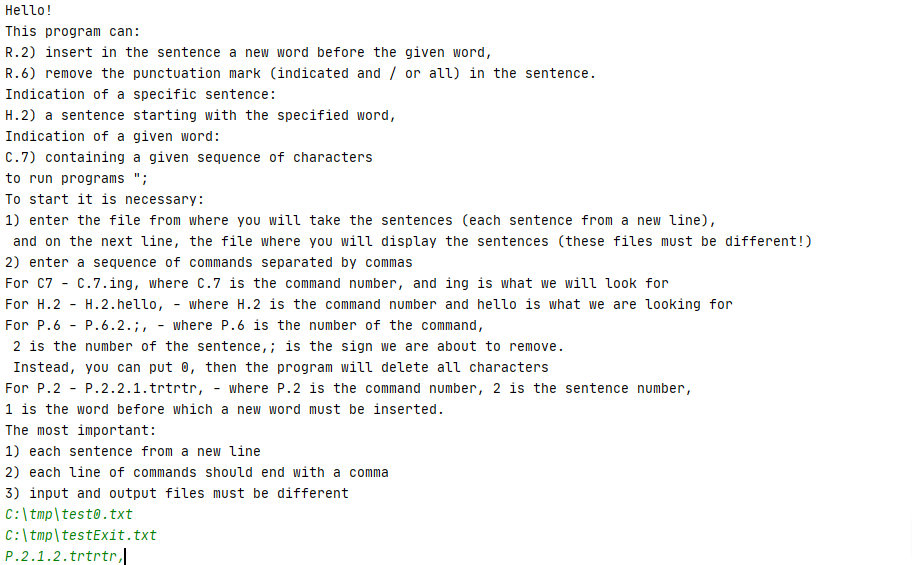
****

****

****

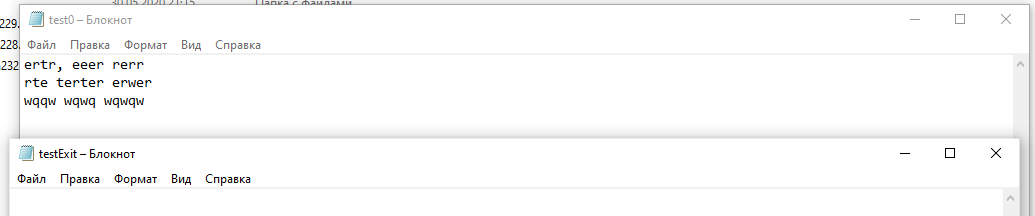
**Примеры работы.**

**запуск и входные данные**

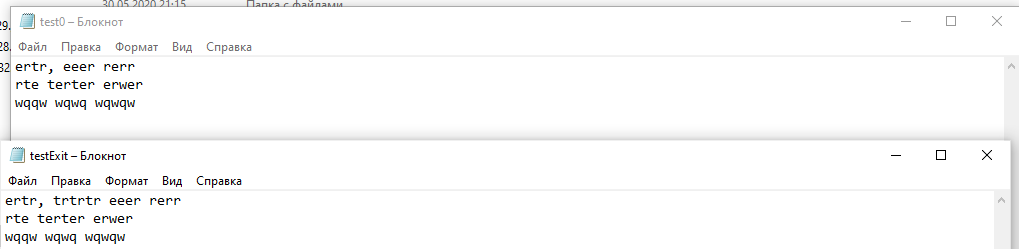
****

**Проверка, что в 1 предложении перед 2 словом ставится указанный набор символов**

**первоначальные данные в файлах для ввода и вывода**

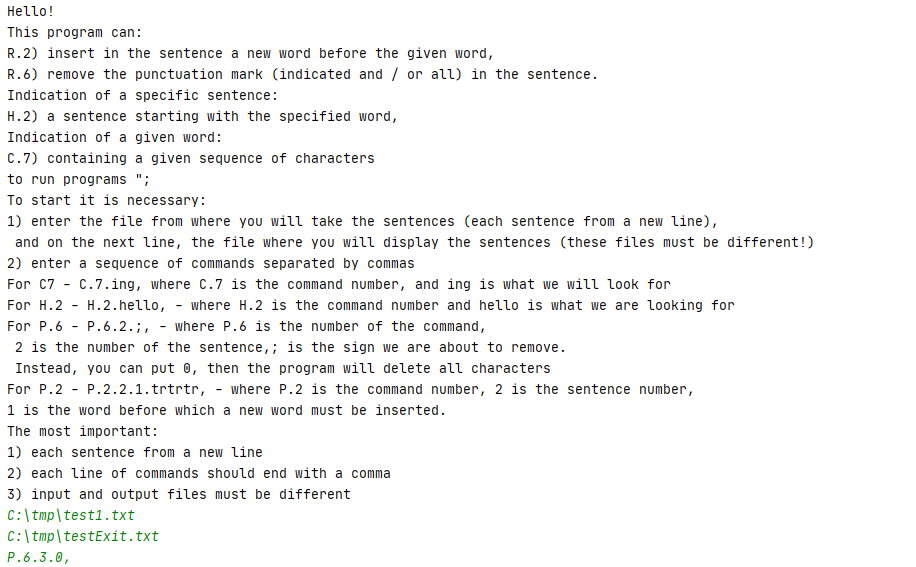
****

**Вывод**

****

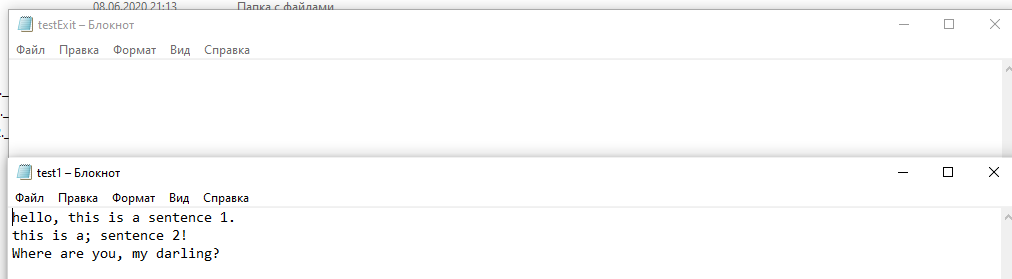
**пример 2**

**Приветствие и ввод**

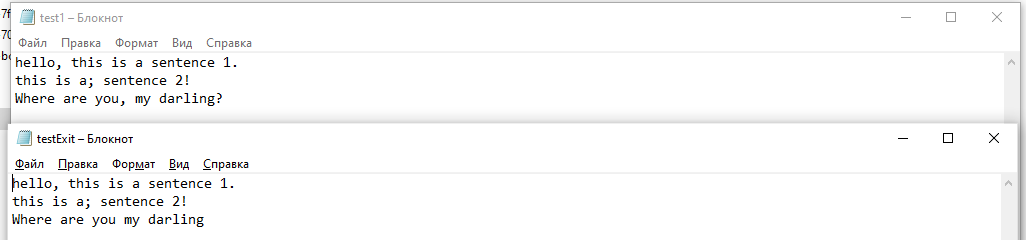
****

**В 3 предложении удаляю все знаки**

**Первоначально в файлах**

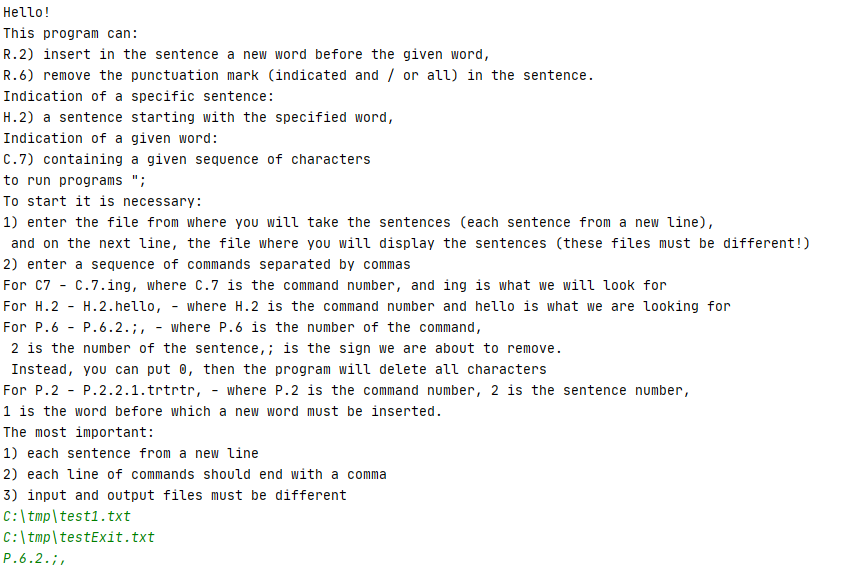
****

**Результат в файлах**

****

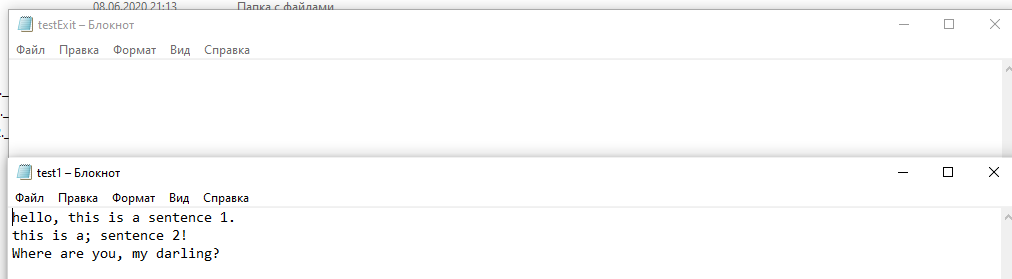
**Пример 3**

**Приветствие и ввод**

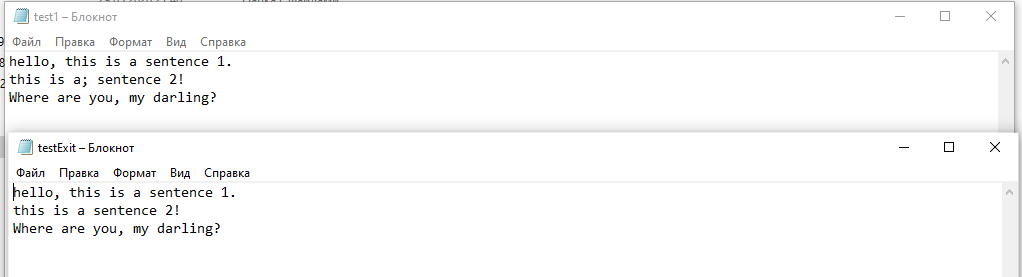
****

**Запрос на удаление из 2 предложения знака “;”**

**первоначально в файлах**

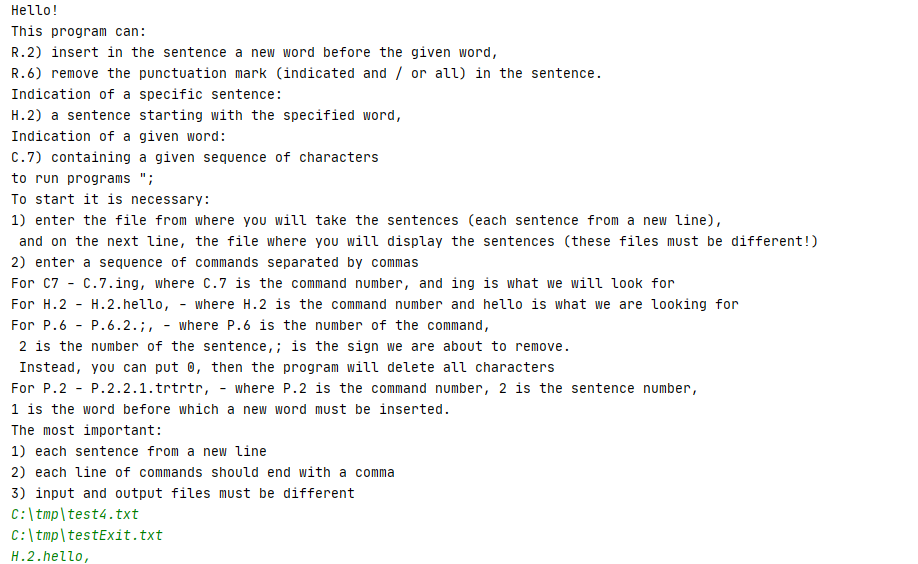
****

**Вывод в файл**

****

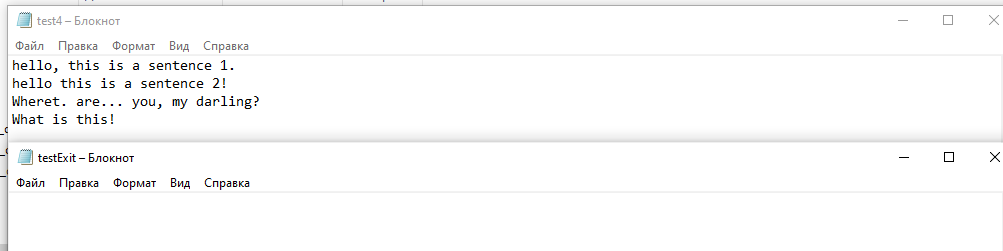
**Пример 4**

**Приветствие и ввод**

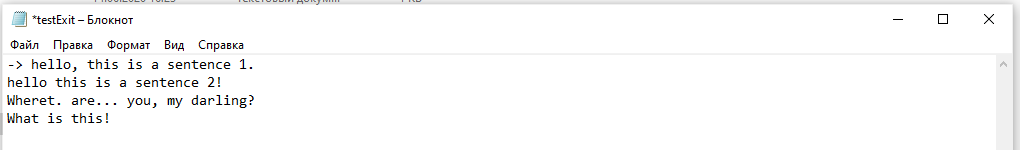
****

**Указать предложение, начинающееся на слово hello**

**первоначально в файлах**

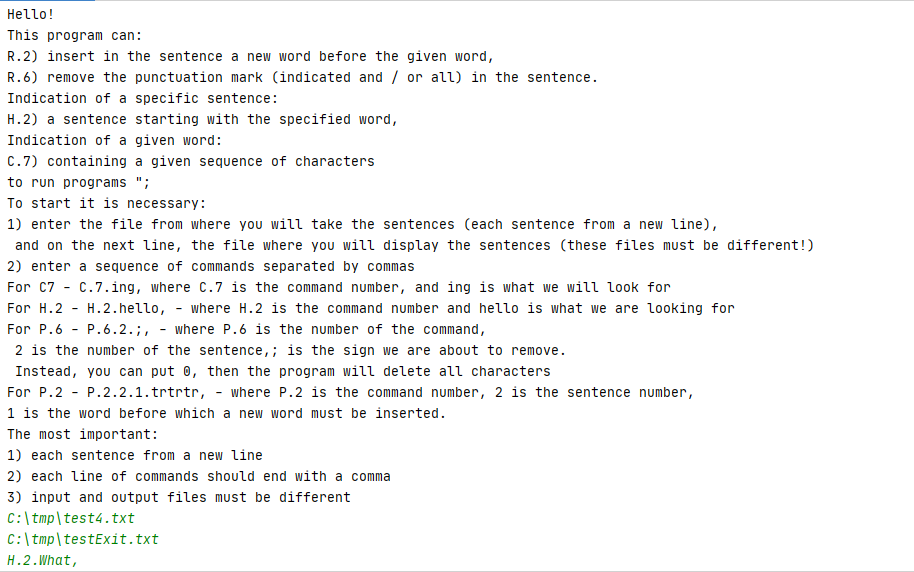
****

**Вывод в файл**

****

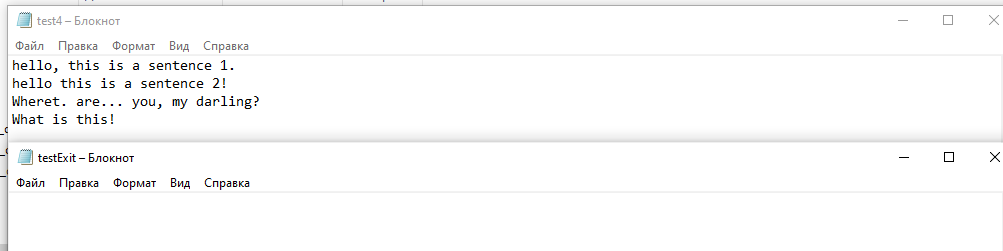
**Пример 5**

**Приветствие и ввод**

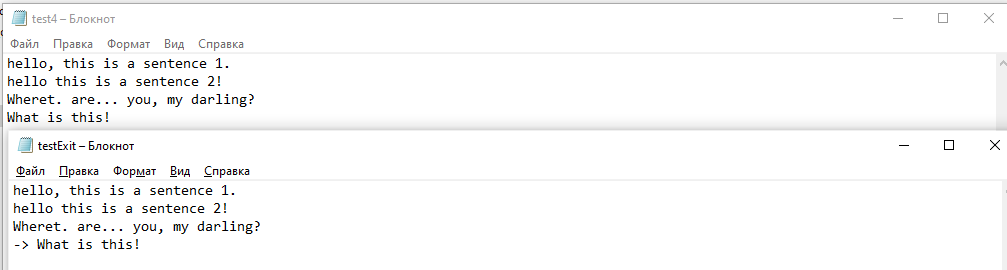
****

**Указать предложение, начинающееся на словa What**

**первоначально в файлах**

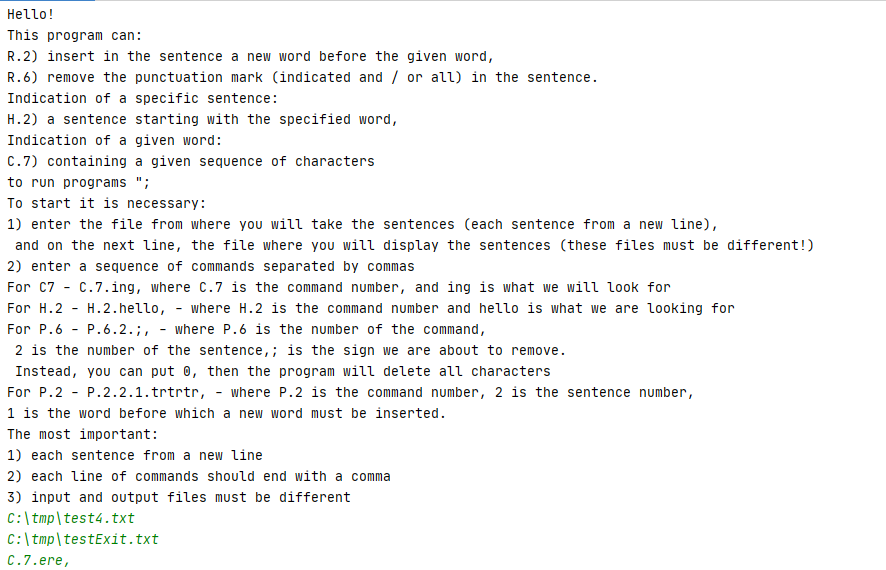
****

**Вывод в файл**

****

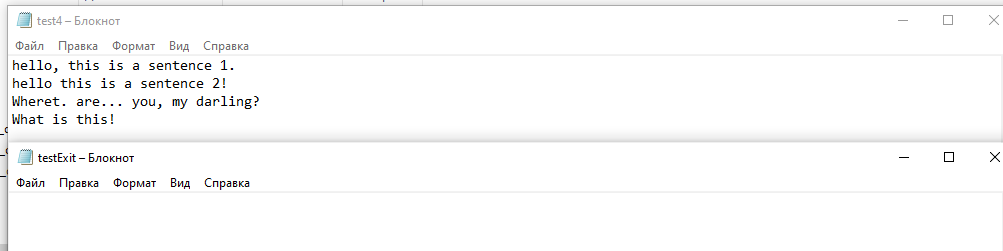
**Пример 6**

**Приветствие и ввод**

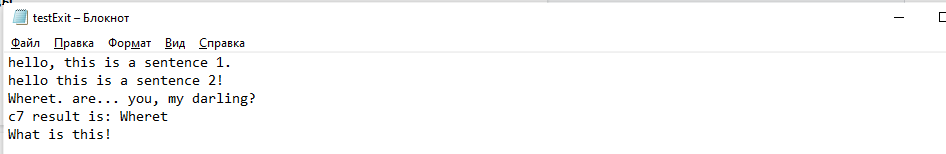
****

**Вернуть слово, в котором встречается указанная последовательность**

**первоначально в файлах**

****

**Вывод в файл**

****

**Выводы.**

В ходе работы над программой были освоены приемы работы с файлами и их библиотеками, служащими для ввода вывода. Получены навыки разработки алгоритмов, а также улучшены навыки работы с линейными односвязными списками.