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).

Purpose of work.

Study and practical development of function programming techniques for representing and performing elementary operations with text (symbolic information) from strings with a marker based on structures (classes) using an array (normal/dynamic) and organizing work with files (reading/writing text) using stdio or fstream libraries, formatting program fragments (types and functions) as a multi-module implementation (a set of files in the project).

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:

C.7) содержащее заданную последовательность символов

Внешний формат хранения данных

Для работы используется 2 файла

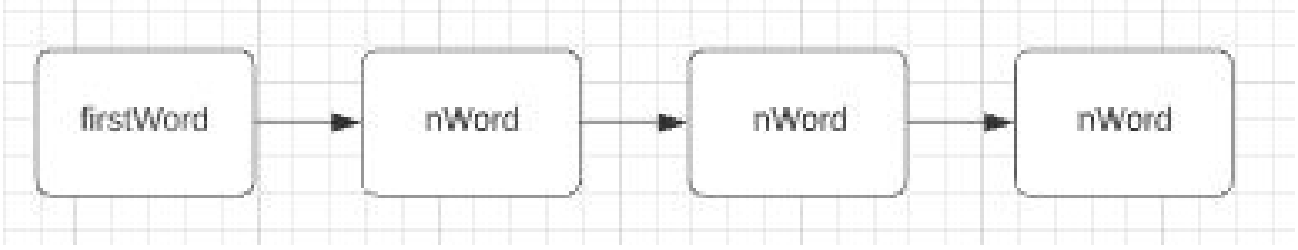
1) для взятия данных
2) для вывода результатов работы

Внутренний формат хранения данных

Вычитанное слово хранится в структуре nexSumb, которая представляет из себя, по сути, linkedList, и каждый элемент хранит указатель на следующий элемент.

```
struct nexSumb  //структура хранения всех слов и символов
{
  nexSumb* nWord;   // указатель на следующее слово
  int length = 0;  // длина слова
  char sent[20];  // само слово
  int flag = 0;   //punctuation checker
  //0 - не пунктуация, 1 - пунктуация, 2 - многоточие
  int deleted = 0;   // 0 - не удалено, 1 - удалено
};
```

схема работы выглядит следующим образом. В firstWord при первом проходе метода splitter помещается ссылка на первый элемент, у которого по полю nWord проходит ссылка на все последующие элементы. Знаки пунктуации и слова считаются за отдельные элементы и обрабатываются отдельно.



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

```
struct MyStor {//for save all inform
  string nameOfFile; // имя файла для печати
```

```
    int sentenceCounter = 0;//порядковый номер предложения
    string commandLine;//line with command   6
    int pos = -1;// P2 position
    int blockerP2 = 0; // блокировка метода P2
    int blockerH2 = 0; // блокировка метода H2
    int blockerC7 = 0; // блокировка метода C7
    int blockerP6 = 0; // блокировка метода P6
    char *newWord = new char[CHAR_MAX];// P2 word
    int nwLength = 0;// p2 newWord length
    char *word = new char[CHAR_MAX];//C7
    int wLength = 0;//c7 word length
};
```

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**	**nameOfFile - имя файла вывода commandLine - строка команд**	**-**

	экземпляров класса Sentence		
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**	**-**	**-**

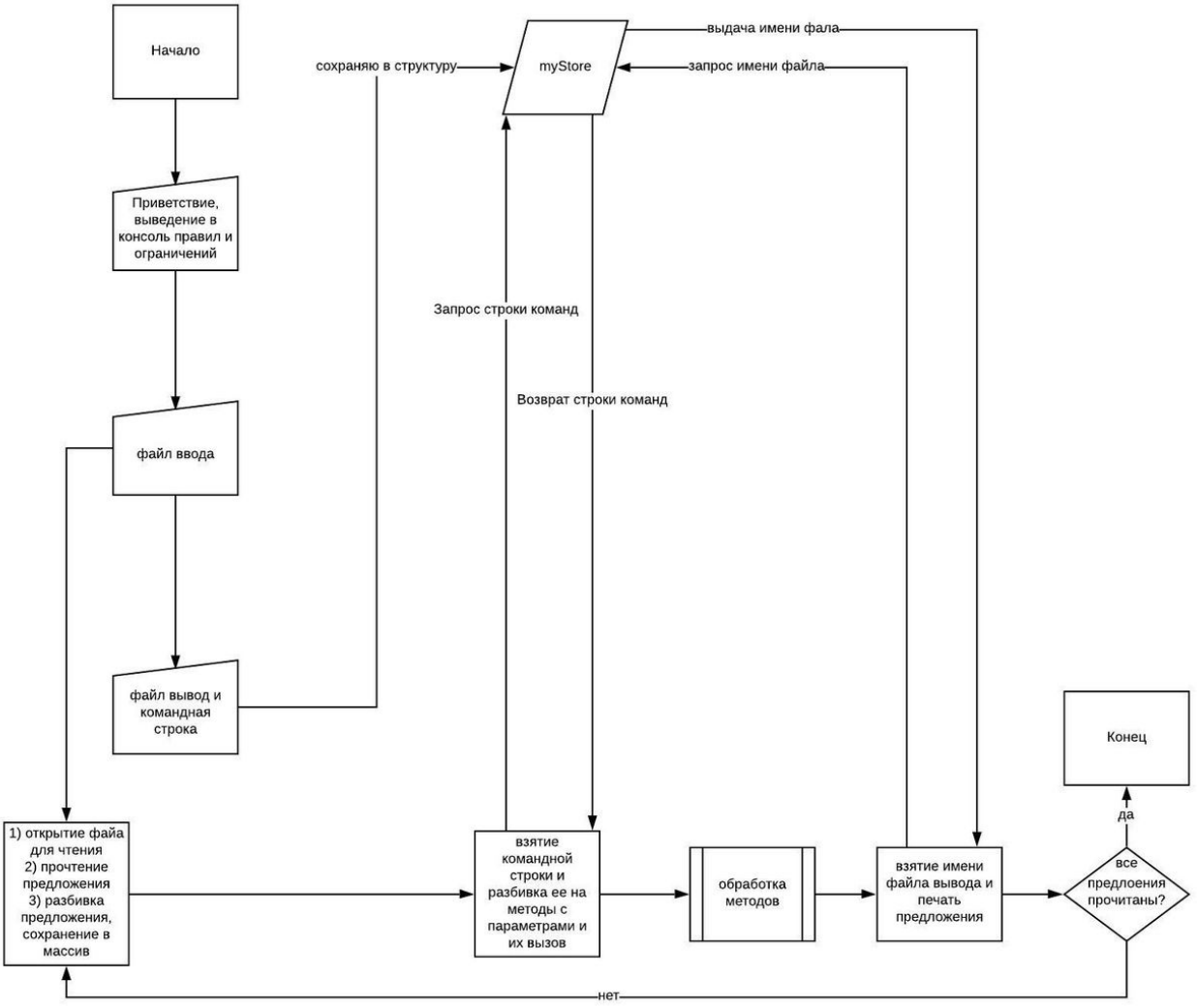
структура nexSumbol()	**хранение всех данных об элементе**	**length - длина слова nWord - указатель на следующее слово(при наличии) charSent[] - слово flag - указатель на то, пунктуация или нет**	

Алгоритм обработки данных

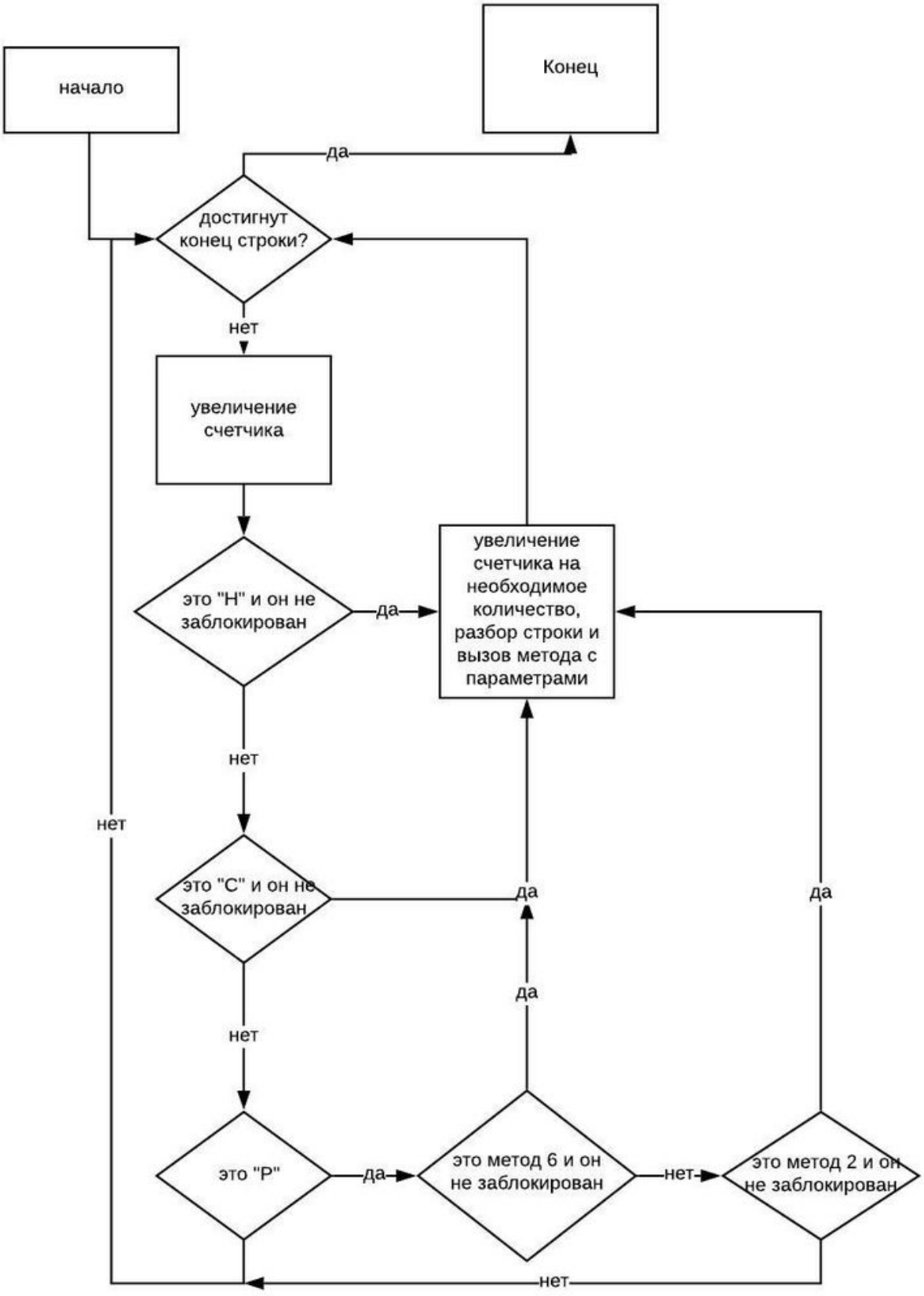
Программа выводит приветствие и описание всех функций, возможностей и правил ввода затем при получении имя файла вывода и строка команд передаются в структуру, а файл ввода открывается для чтения. Для каждой строки создается новый экземпляр класса, в который передается полученная строка, внутри класса 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.

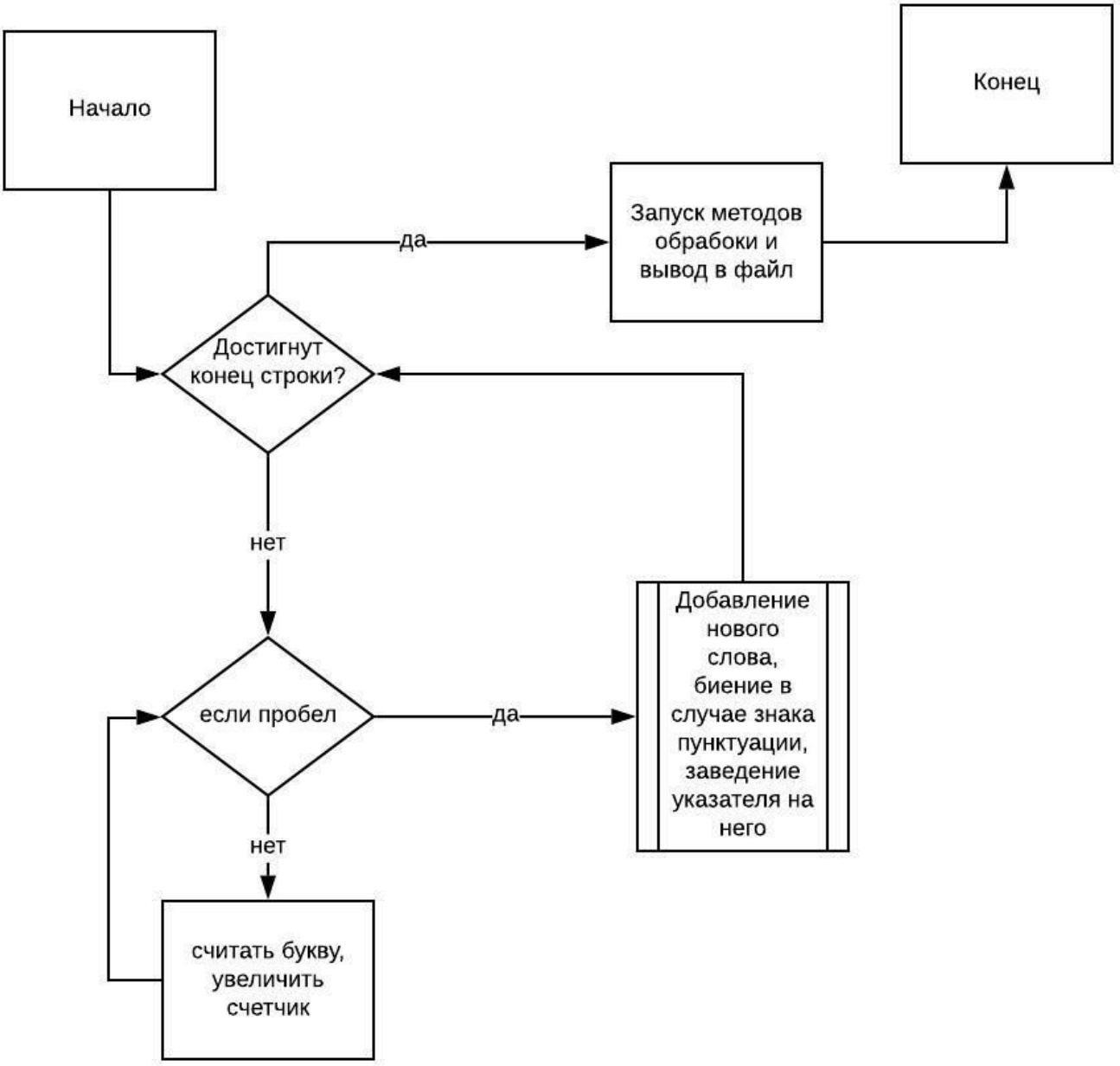
Блок-схема(общая)



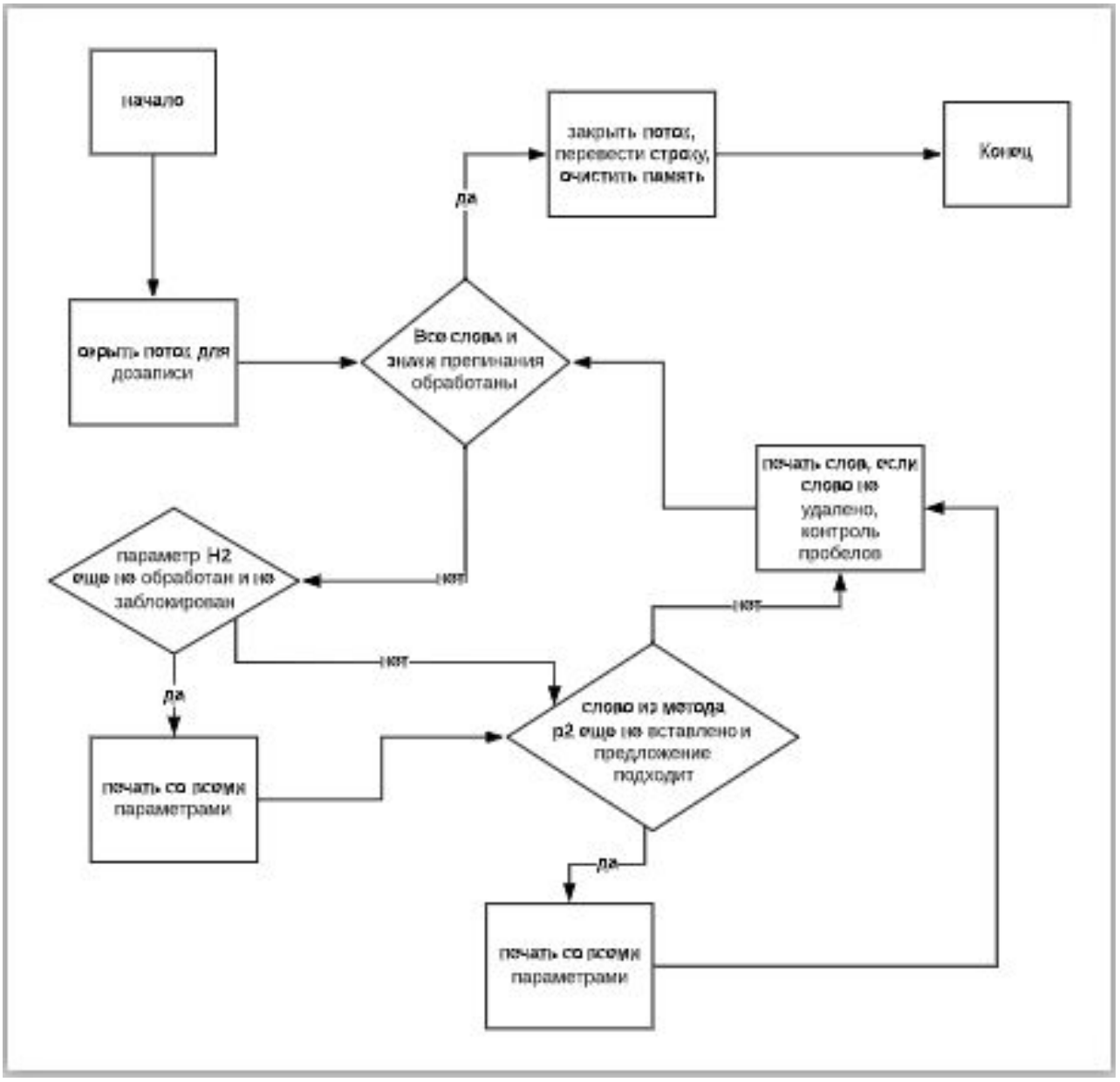
Блок-схема (command parser)



Блок-схема(метод add, метод split)


Начало
Конец
Запуск методов обрабоки и вывод в файл
да
Достигнут конец строки?
нет
Добавление нового слова, биение в случае знака пунктуации, заведение указателя на него
если пробел
да
нет
считать букву, увеличить счетчик

Блок-схема(метод printer)



Код программы

```
#include <iostream>
#include <fstream>
#include <string>
#include <climits>

using namespace std;

struct MyStor {//for save all inform
    string nameOfFile;
    int sentenceCounter = 0;//порядковый номер предложения
    string commandLine;//line with command   6
    int pos = -1;// P2 position
    int blockerP2 = 0;
    int blockerH2 = 0;
    int blockerC7 = 0;
    int blockerP6 = 0;
    char *newWord = new char[CHAR_MAX];// P2 word
    int nwLength = 0;// p2 newWord length
    char *word = new char[CHAR_MAX];//C7
    int wLength = 0;//c7 word length
};

struct nexSumb//структура хранения всех слов и символов
{
    nexSumb* nWord;// указатель на следующее слово

    int length = 0;// длина слова
    char sent[20];// само слово
    int flag = 0;//punctuation checker
    //0 - не пунктуация, 1 - пунктуаця, 2 - многоточие
    int deleted = 0; // 0 - не удалено, 1 - удалено
};

MyStor myStor;


class Sentence//новое предложение
{
public:
    int sWCounter = 0;// подсчет количества элементов в sentenceWith
    nexSumb *firstWord; // указатель на первое слово

    //для создания нового предложения просто ввожу строку
    Sentence(string line) { splitter(line); }

    //метод парсит предложение на слова и вызывает метод добавления
    void splitter(string line) {
        int totalWordCount = 0;// количество букв в слове, на один меньше для записи в массив
        int i = 0;
        char *wordFast = new char [line.length()];
        while (line.length() > i && line[i] != ' ')
        {
```

```
            wordFast[i] = line[i];
            i++;
            totalWordCount++;
        }
        nexSumb *tmp = add(wordFast,totalWordCount);
        nexSumb *prevWord = new nexSumb;
        if(sWCounter == 2)prevWord = tmp->nWord;
        else prevWord = firstWord;
        int j = totalWordCount + 1;// корректировка на пробел
        totalWordCount = 0;
        for (int i = j; i < line.length() + 1; ++i) {
            if (line[i] == ' ' || i == line.length())
            {
                nexSumb *temp;
                int controller = sWCounter;
                temp = add(wordFast, totalWordCount);// связываю его с предыдущим элементом
                prevWord->nWord = temp;
                controller +=2;
                if(controller  == sWCounter) prevWord = temp->nWord;
                else prevWord = temp; // элемент становится предыдущим
                totalWordCount = 0;
            }
            else {
                wordFast[totalWordCount] = line[i];
                totalWordCount++;
            }

        }
        commandParser();
        strPrinter();
        delete[] wordFast;
}

// метод добавляет новое слово в структуру, если была пунктуация, то добавится и она отдельным словом. Возвращает
// указатель на это слово
nexSumb *add(char *wordAdd, int length) {
    int controller = sWCounter;
    int fastLength = length;// хранение первоначального размера
    nexSumb *sumb = NULL;// знак препинания если будет
    //если последняя буква в слове - знак препинания, делаю длину слова меньше на 1

    if (punctuationChecker( mark: wordAdd[length - 1]) || wordAdd[length - 1] == '.') {
        if (wordAdd[length - 2] == '.')// проверяю на многоточие
        {
            nexSumb *nexSumb = new struct nexSumb;
            nexSumb->flag = 2;// многоточие
            nexSumb->length = 3;
            nexSumb->sent[0] = '.';
            nexSumb->sent[1] = '.';
            nexSumb->sent[2] = '.';
            sumb = nexSumb;
            length -= 3;
            sWCounter++;
```

```
        } else { // если не многоточие, добавляю
            nexSumb *nexSumb = new struct nexSumb;
            nexSumb->flag = 1;
            nexSumb->length = 1;
            nexSumb->sent[0] = wordAdd[length - 1];
            sumb = nexSumb;
            length -= 1;
            sWCounter++;
        }
    }//если была пунктуация я создал новую структуру
    nexSumb *word = new nexSumb;
    for (int j = 0; j < length; ++j) {
        word->sent[j] = wordAdd[j];
    }// основное добавление слова
    sWCounter++;
    word->length = length;
    if (length < fastLength) //проверяю, был ли в конце знак препинания и добавляю в слово как следующий элемент
    {
        word->nWord = sumb;
    }
    if(controller == 0) firstWord = word;
    return word;
}
```

```
// check punctuation mark
// if the char is a punctuation mark function return true, and return false if is not
static bool punctuationChecker(char mark) {
    switch (mark) {
        case ',':
            return true;
        case ':':
            return true;
        case ';':
            return true;
        case '-':
            return true;
        case '=':
            return true;
        case '_':
            return true;
        case '*':
            return true;
        case '(':
            return true;
        case ')':
            return true;
        case '{':
```

```
                return true;
            case '}':
                return true;
            case '[':
                return true;
            case ']':
                return true;
            case '/':
                return true;
            case '|':
                return true;
            case '\\':
                return true;
            case '+':
                return true;
            case '@':
                return true;
            case '#':
                return true;
            case '$':
                return true;
            case '%':
                return true;
            case '^':
                return true;

            case '!':
                return true;//возможно вынести в отдельную проверку
            case '?':
                return true;//возможно вынести в отдельную проверку
            default:
                return false;
        }
    }

// в этом методе буду разбивать строку комманд и сразу вызывать связанные методы
    void commandParser() {
        for (int i = 0; i < myStor.commandLine.length(); ++i) {
            int length = 0;
            int tempI;
            if (myStor.commandLine[i] == 'H' && myStor.blockerH2 == 0) {
                tempI = i;
                i += 4;//+ 2 для пропуска номера и точки
                while (myStor.commandLine[i] != ',') {
                    length++;
                    i++;
                }
```

```
            i = tempI;
            i +=4;
            char *nxWord = new char[length];
            length = 0;
            while (myStor.commandLine[i] != ',') {
                nxWord[length] = myStor.commandLine[i];
                length++;
                i++;
            }
            startOnThisWord(nxWord, length);
            delete[] nxWord;
        }
        if (myStor.commandLine[i] == 'C' && myStor.blockerC7 == 0) {
            tempI = i;
            i += 4;//
            while (myStor.commandLine[i] != ',') {
                length++;
                i++;
            }
            i = tempI;
            i += 4;
            char *nxWord = new char[length];
            length = 0;
            while (myStor.commandLine[i] != ',') {
                nxWord[length] = myStor.commandLine[i];
                length++;
                i++;
            }
            usingSumbolSequence(nxWord, length);
            myStor.nwLength = length;
        }
        if (myStor.commandLine[i] == 'P') {
            int index = (int) myStor.commandLine[i + 4] - 48;
            if (index == myStor.sentenceCounter) {
                if (myStor.commandLine[i + 2] == '6' && myStor.blockerP6 == 0) {
                    i += 6;
                    if (myStor.commandLine[i] == '0') {
                        i += 1;
                        deletePunctuationMark( isDeleteAll: 0,  punctMarc: reinterpret_cast<char *>('r'),  length: 0);
                    } else {
                        if (myStor.commandLine[i] != '.') {
                            char *punctu = &myStor.commandLine[i];
                            deletePunctuationMark( isDeleteAll: 1, punctu,  length: 1);
                            i += 2;
                        } else {
                            char *punctu = new char[3]{'.','.','.'};
                            deletePunctuationMark( isDeleteAll: 1, punctu,  length: 3);
                            i += 2;
                        }
```

```
                }
            } else if (myStor.blockerP2 == 0){//&& blockerP2 == 0
                i += 4;
                myStor.pos = myStor.commandLine[i + 2] - 48;// позиция
                i += 2;
                length = 0;
                while (myStor.commandLine[i + 2] != ',') {
                    myStor.newWord[length] = myStor.commandLine[i + 2];
                    length++;
                    i++;
                }
                myStor.nwLength = length;
            }
        }
    }
}


// / isDeleteAll - если не 0 то удалить только punctMarc,
// иначе удалить все, модифицирует punctuation массив, заменяя выбранные знаки пробелом
void deletePunctuationMark(int isDeleteAll, char *punctMarc, int length) {
    nexSumb *nextSumb = firstWord->nWord;// ссылаю сразу на 2 элемент, если что удаляю его
   if(isDeleteAll == 1)
   {
       if(firstWord->flag != 0 && length == firstWord->length)
        if(firstWord->sent[0] == '.' || firstWord->sent[0] == *punctMarc)
        {
            firstWord->deleted = 1;// удалил знак
        }
    for (int i = 1; i < sWCounter; ++i)
    {// прохожу по всем элементам в
        if(nextSumb->flag != 0 && length == nextSumb->length)
            if(nextSumb->sent[0] == *punctMarc)
            {
                nextSumb->deleted = 1;// удалил знак
            }
        nexSumb *temp = nextSumb->nWord;
        nextSumb = temp;
    }
   }
   else
   {
       if(firstWord->flag != 0) firstWord->deleted = 1;
```

```
        for (int i = 1; i < sWCounter; ++i)
        {// прохожу по всем элементам в
            if(nextSumb->flag != 0 )nextSumb->deleted = 1;// удалил знак
            nexSumb *temp = nextSumb->nWord;
            nextSumb = temp;
        }
    }
    myStor.blockerP6 = 1;
}
void startOnThisWord(char *word, int length) {
    if (firstWord->length == length) {
        int controller = 0;
        for (int i = 0; i < firstWord->length; ++i) {
            if (firstWord->sent[i] != word[i]) break;
            else controller++;
        }
        if (controller == length) {
            myStor.blockerH2 = 1;
        }
    }
}//H2

void usingSumbolSequence(char *wordUse, int length) {
    int counter = 0;

    nexSumb *nextSumb = firstWord->nWord;// ссылаю сразу на 2 элемент, если что удаляю его
    if(firstWord->flag == 0 && length <= firstWord->length)
        {
            for (int i = 0; i < firstWord->length; ++i) {
                if (firstWord->sent[i] == wordUse[counter]) counter++;
                else if(length == counter) break;
                else counter = 0;
            }
            if (counter == length) {
                myStor.word = firstWord->sent;
                myStor.blockerC7 = -1;
                myStor.wLength = firstWord->length;
                return;
            }
        }

    for (int i = 1; i < sWCounter; ++i)
    {// прохожу по всем элементам
        if(nextSumb->flag == 0 && length <= nextSumb->length)
            {

                for (int j = 0; j < nextSumb->length; ++j) {
                    if (nextSumb->sent[j] == wordUse[counter]) counter++;
                    else if(length == counter) break;//если подошло
```

```
                    else counter = 0;
                }
            }
        nexSumb *temp = nextSumb->nWord;
        nextSumb = temp;
    }

    if (counter == length) {
        myStor.word = firstWord->sent;
        myStor.blockerC7 = -1;
        myStor.wLength = firstWord->length;
        return;
    }

}

// здесь я возвращаю адрес первого вхождения указанного слова
void strPrinter() {
    int counter = 1; // подсчет вывода слов для пункта P2
    ofstream out;// поток для записи
    nexSumb *nextSumb = firstWord;// следующий элемент
    out.open(myStor.nameOfFile, ios::app);// окрываем файл для записи
    if (out.is_open()) {
        for (int i = 0; i < sWCounter; ++i) {

            if (myStor.blockerH2 == 1) {
                out << "->" << " ";
                myStor.blockerH2 = -1;
            }
            if (myStor.pos == counter && myStor.blockerP2 == 0)//проверка для пункта p2
            {
                if (nextSumb->flag == 0) {
                    if (myStor.pos - 1 != 0) out << " ";
                    for (int j = 0; j < myStor.nwLength; ++j) {
                        out << myStor.newWord[j];
                    }
                    if (myStor.pos - 1 == 0) out << " ";
                    myStor.blockerP2 = 1;//блокирую метод p2
                }
            }

            if (nextSumb->deleted == 0) // слово не удалено
                {
                    if (nextSumb->flag == 0) {
                        if (i != 0)out << " ";// если не первое слово или не пунктуация выведу пробел
                        counter++;
                    }

                    for (int j = 0; j < nextSumb->length; ++j) {
```

```
                    out << nextSumb->sent[j];
                }
            }

        nexSumb *temp = nextSumb->nWord;
        nextSumb = temp;
    }

        }
        out << '\n';
    }
};

int main() {
    string nameOfFile;
    cout << "Hello!\n"
            "This program can:\n"
            "R.2) insert in the sentence a new word before the given word,\n"
            "R.6) remove the punctuation mark (indicated and / or all) in the sentence.\n"
            "Indication of a specific sentence:\n"
            "H.2) a sentence starting with the specified word,\n"
            "Indication of a given word:\n"
            "C.7) containing a given sequence of characters\n"
            "to run programs \";\n"
            "To start it is necessary:\n"
            "1) enter the file from where you will take the sentences (each sentence from a new line),\n"
            " and on the next line, the file where you will display the sentences (these files must be different!)\n"
            "2) enter a sequence of commands separated by commas\n"
            "For C7 - C.7.ing, where C.7 is the command number, and ing is what we will look for\n"
            "For H.2 - H.2.hello, - where H.2 is the command number and hello is what we are looking for\n"
            "For P.6 - P.6.2.;, - where P.6 is the number of the command,\n"
            " 2 is the number of the sentence,; is the sign we are about to remove.\n"
            " Instead, you can put 0, then the program will delete all characters\n"
            "For P.2 - P.2.2.1.trtrtr, - where P.2 is the command number, 2 is the sentence number,\n"
            "1 is the word before which a new word must be inserted.\n"
            "The most important:\n"
            "1) each sentence from a new line\n"
            "2) each line of commands should end with a comma\n"
            "3) input and output files must be different\n";
    cin >> nameOfFile; // where i take text
    cin >> myStor.nameOfFile; // where i write answer
    cin >> myStor.commandLine;
    ifstream in;
    in.open(nameOfFile);
    string line;
    if (in.is_open()) {
        while (getline( & in,  & line)) {
            myStor.sentenceCounter++;
            new Sentence(line);
            if (myStor.blockerC7 == -1) {
```

```
                ofstream out;               // поток для записи
                out.open(myStor.nameOfFile, ios::app);// окрываем файл для записи
                if (out.is_open()) {
                    out << "c7 result is: ";
                    for (int j = 0; j < myStor.wLength; ++j) {
                        out << myStor.word[j];
                    }
                    out << '\n';
                    myStor.blockerC7 = 1;
                    delete [] myStor.word;
                }
                out.close();
            }
        }
    }
    in.close();
}
```

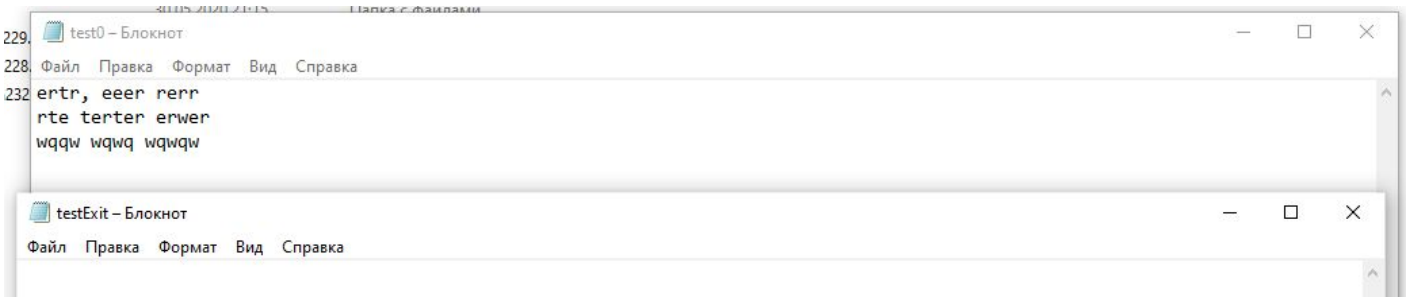
Примеры работы.

запуск и входные данные

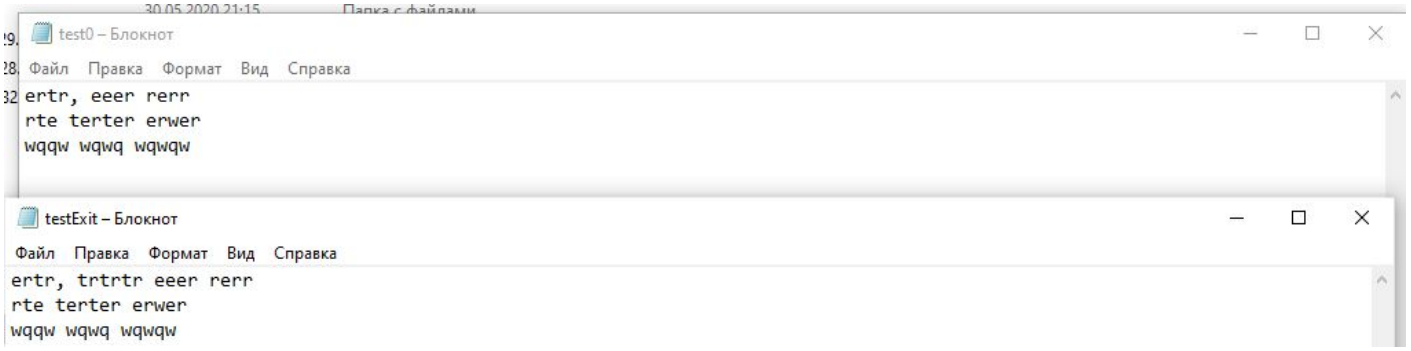
```
Hello!
This program can:
R.2) insert in the sentence a new word before the given word,
R.6) remove the punctuation mark (indicated and / or all) in the sentence.
Indication of a specific sentence:
H.2) a sentence starting with the specified word,
Indication of a given word:
C.7) containing a given sequence of characters
to run programs ";
To start it is necessary:
1) enter the file from where you will take the sentences (each sentence from a new line),
 and on the next line, the file where you will display the sentences (these files must be different!)
2) enter a sequence of commands separated by commas
For C7 - C.7.ing, where C.7 is the command number, and ing is what we will look for
For H.2 - H.2.hello, - where H.2 is the command number and hello is what we are looking for
For P.6 - P.6.2.;, - where P.6 is the number of the command,
 2 is the number of the sentence,; is the sign we are about to remove.
 Instead, you can put 0, then the program will delete all characters
For P.2 - P.2.2.1.trtrtr, - where P.2 is the command number, 2 is the sentence number,
1 is the word before which a new word must be inserted.
The most important:
1) each sentence from a new line
2) each line of commands should end with a comma
3) input and output files must be different
C:\tmp\test0.txt
C:\tmp\testExit.txt
P.2.1.2.trtrtr,
```

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

первоначальные данные в файлах для ввода и вывода



Вывод



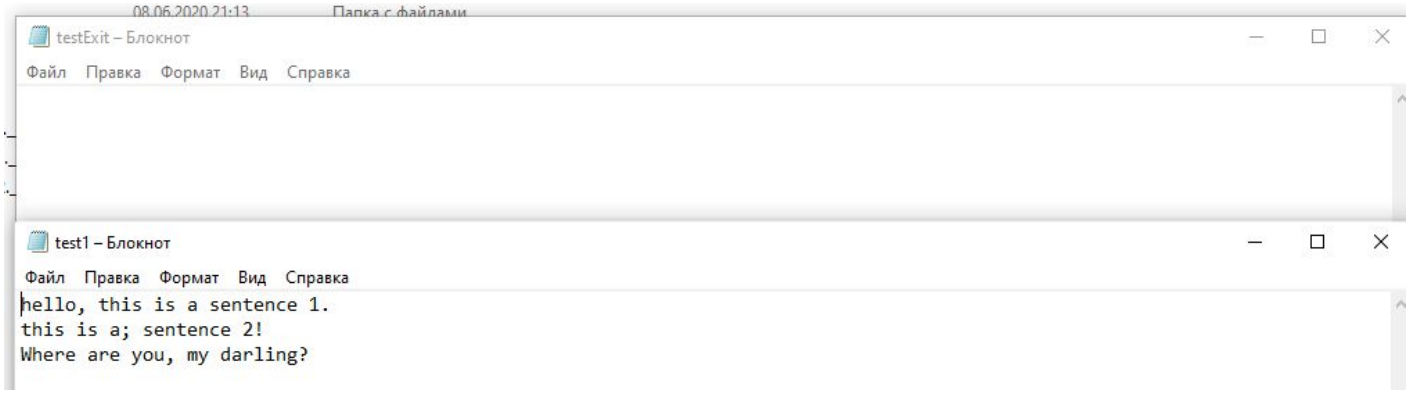
пример 2

Приветствие и ввод

```
Hello!
This program can:
R.2) insert in the sentence a new word before the given word,
R.6) remove the punctuation mark (indicated and / or all) in the sentence.
Indication of a specific sentence:
H.2) a sentence starting with the specified word,
Indication of a given word:
C.7) containing a given sequence of characters
to run programs ";
To start it is necessary:
1) enter the file from where you will take the sentences (each sentence from a new line),
 and on the next line, the file where you will display the sentences (these files must be different!)
2) enter a sequence of commands separated by commas
For C7 - C.7.ing, where C.7 is the command number, and ing is what we will look for
For H.2 - H.2.hello, - where H.2 is the command number and hello is what we are looking for
For P.6 - P.6.2.;, - where P.6 is the number of the command,
 2 is the number of the sentence,; is the sign we are about to remove.
 Instead, you can put 0, then the program will delete all characters
For P.2 - P.2.2.1.trtrtr, - where P.2 is the command number, 2 is the sentence number,
1 is the word before which a new word must be inserted.
The most important:
1) each sentence from a new line
2) each line of commands should end with a comma
3) input and output files must be different
C:\tmp\test1.txt
C:\tmp\testExit.txt
P.6.3.0,
```

В 3 предложении удаляю все знаки

Первоначально в файлах



Результат в файлах

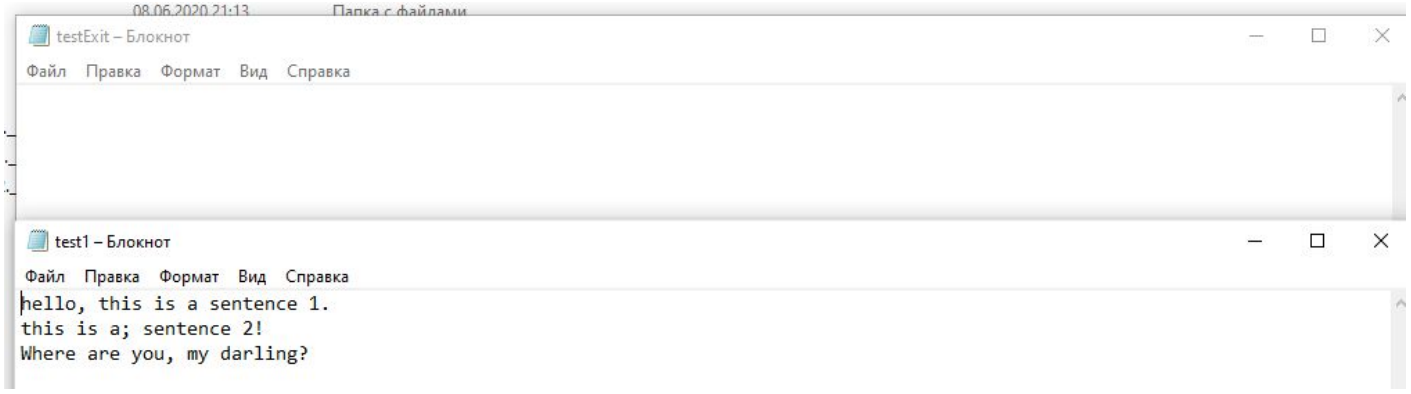
Пример 3

Приветствие и ввод

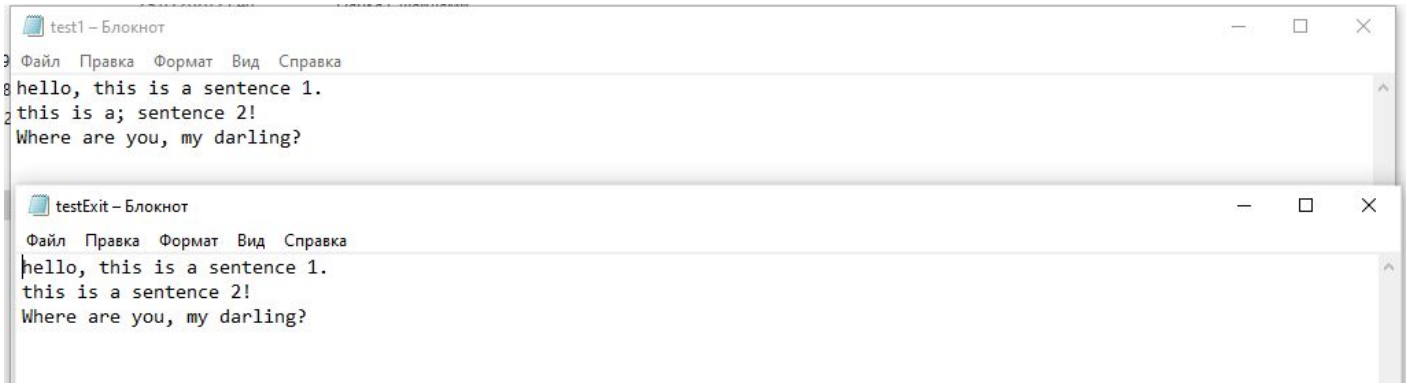
```
Hello!
This program can:
R.2) insert in the sentence a new word before the given word,
R.6) remove the punctuation mark (indicated and / or all) in the sentence.
Indication of a specific sentence:
H.2) a sentence starting with the specified word,
Indication of a given word:
C.7) containing a given sequence of characters
to run programs ";
To start it is necessary:
1) enter the file from where you will take the sentences (each sentence from a new line),
 and on the next line, the file where you will display the sentences (these files must be different!)
2) enter a sequence of commands separated by commas
For C7 - C.7.ing, where C.7 is the command number, and ing is what we will look for
For H.2 - H.2.hello, - where H.2 is the command number and hello is what we are looking for
For P.6 - P.6.2.;, - where P.6 is the number of the command,
 2 is the number of the sentence,; is the sign we are about to remove.
 Instead, you can put 0, then the program will delete all characters
For P.2 - P.2.2.1.trtrtr, - where P.2 is the command number, 2 is the sentence number,
1 is the word before which a new word must be inserted.
The most important:
1) each sentence from a new line
2) each line of commands should end with a comma
3) input and output files must be different
C:\tmp\test1.txt
C:\tmp\testExit.txt
P.6.2.;,
```

Запрос на удаление из 2 предложения знака “;”

первоначально в файлах



Вывод в файл



Пример 4

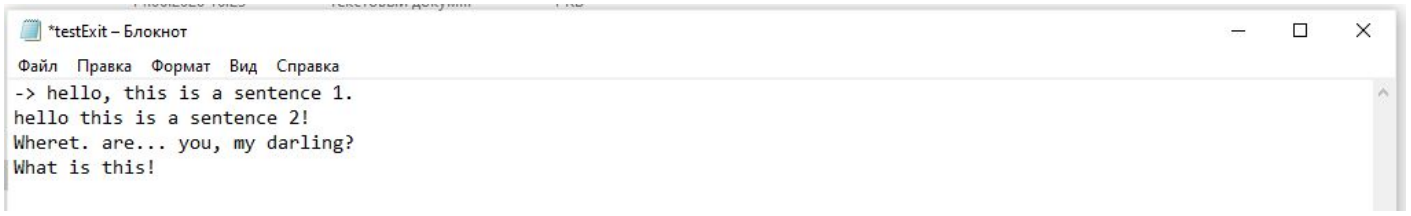
Приветствие и ввод

```
Hello!
This program can:
R.2) insert in the sentence a new word before the given word,
R.6) remove the punctuation mark (indicated and / or all) in the sentence.
Indication of a specific sentence:
H.2) a sentence starting with the specified word,
Indication of a given word:
C.7) containing a given sequence of characters
to run programs ";
To start it is necessary:
1) enter the file from where you will take the sentences (each sentence from a new line),
 and on the next line, the file where you will display the sentences (these files must be different!)
2) enter a sequence of commands separated by commas
For C7 - C.7.ing, where C.7 is the command number, and ing is what we will look for
For H.2 - H.2.hello, - where H.2 is the command number and hello is what we are looking for
For P.6 - P.6.2.;, - where P.6 is the number of the command,
 2 is the number of the sentence,; is the sign we are about to remove.
 Instead, you can put 0, then the program will delete all characters
For P.2 - P.2.2.1.trtrtr, - where P.2 is the command number, 2 is the sentence number,
1 is the word before which a new word must be inserted.
The most important:
1) each sentence from a new line
2) each line of commands should end with a comma
3) input and output files must be different
C:\tmp\test4.txt
C:\tmp\testExit.txt
H.2.hello,
```

Указать предложение, начинающееся на слово hello

первоначально в файлах

Вывод в файл



Пример 5

Приветствие и ввод

```
Hello!
This program can:
R.2) insert in the sentence a new word before the given word,
R.6) remove the punctuation mark (indicated and / or all) in the sentence.
Indication of a specific sentence:
H.2) a sentence starting with the specified word,
Indication of a given word:
C.7) containing a given sequence of characters
to run programs ";
To start it is necessary:
1) enter the file from where you will take the sentences (each sentence from a new line),
 and on the next line, the file where you will display the sentences (these files must be different!)
2) enter a sequence of commands separated by commas
For C7 - C.7.ing, where C.7 is the command number, and ing is what we will look for
For H.2 - H.2.hello, - where H.2 is the command number and hello is what we are looking for
For P.6 - P.6.2.;, - where P.6 is the number of the command,
 2 is the number of the sentence,; is the sign we are about to remove.
 Instead, you can put 0, then the program will delete all characters
For P.2 - P.2.2.1.trtrtr, - where P.2 is the command number, 2 is the sentence number,
1 is the word before which a new word must be inserted.
The most important:
1) each sentence from a new line
2) each line of commands should end with a comma
3) input and output files must be different
C:\tmp\test4.txt
C:\tmp\testExit.txt
H.2.What,
```

Указать предложение, начинающееся на слова What

первоначально в файлах

Вывод в файл

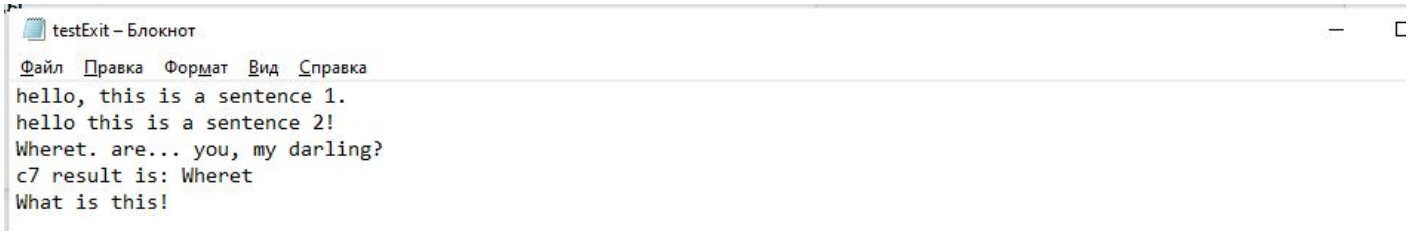
Пример 6

Приветствие и ввод

```
Hello!
This program can:
R.2) insert in the sentence a new word before the given word,
R.6) remove the punctuation mark (indicated and / or all) in the sentence.
Indication of a specific sentence:
H.2) a sentence starting with the specified word,
Indication of a given word:
C.7) containing a given sequence of characters
to run programs ";
To start it is necessary:
1) enter the file from where you will take the sentences (each sentence from a new line),
 and on the next line, the file where you will display the sentences (these files must be different!)
2) enter a sequence of commands separated by commas
For C7 - C.7.ing, where C.7 is the command number, and ing is what we will look for
For H.2 - H.2.hello, - where H.2 is the command number and hello is what we are looking for
For P.6 - P.6.2.;, - where P.6 is the number of the command,
 2 is the number of the sentence,; is the sign we are about to remove.
 Instead, you can put 0, then the program will delete all characters
For P.2 - P.2.2.1.trtrtr, - where P.2 is the command number, 2 is the sentence number,
1 is the word before which a new word must be inserted.
The most important:
1) each sentence from a new line
2) each line of commands should end with a comma
3) input and output files must be different
C:\tmp\test4.txt
C:\tmp\testExit.txt
C.7.ere,
```

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.