МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка текстовых данных

Студент гр. 2300 _____ Войнов А.Н.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2022

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Войнов А.Н.

Группа 2300

Тема работы: Обработка текстовых данных

Исходные данные:

Вариант 18

Программе на вход подается текст (текст представляет собой предложения, разделенные точкой. Предложения - набор слов, разделенные пробелом или запятой, слова - набор латинских букв и цифр. Длина текста и каждого предложения заранее не известна.

Программа должна сохранить этот текст в динамический массив строк и оперировать далее только с ним.

Программа должна найти и удалить все повторно встречающиеся предложения (сравнивать их следует посимвольно, но без учета регистра).

Далее, программа должна запрашивать у пользователя одно из следующих доступных действий (программа должна печатать для этого подсказку. Также следует предусмотреть возможность выхода из программы):

1) Удалить все символы в начале и конце предложения так, чтобы в итоге первый и последний символ предложения были различными (без учета регистра). Например, предложение “abcdba” должно принять вид “cd”.
2) Отсортировать все слова в предложении в лексикографическом порядке.
3) Удалить все предложения, в которых хотя бы одного слово является

палиндромом.

4) Вывести все предложения, в которых есть слово “HiddenAgent” и которое не является первым словом.

Все сортировки должны осуществляться с использованием функции стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания: 20.10.2022

Дата сдачи реферата: 20.12.2022

Дата защиты реферата: 22.12.2022

Студент _____ Войнов А.Н

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

Курсовая работа представляет собой программу на языке СИ, написанную с использованием стандартной библиотеки языка.

Программа считывает текст из консоли, затем удаляет повторяющиеся предложения в тексте и предлагает пользователю варианты обработки текста или закрыть приложение. Далее программа обрабатывает и выводит текст, согласно условию задания. Затем пользователь может повторить обработку текста или завершить работу программы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: Разработать программу, которая считывает текст, обрабатывает его согласно условию и выводит.

Для достижения цели требуется решить следующие задачи:

1) Изучить функции стандартной библиотеки.
2) Реализовать ввод и вывод текст, сохранение его в динамический массив.
3) Реализовать функции обработки текста.
4) Протестировать программу

ХОД РАБОТЫ

2.1 Структуры

Реализованы 3 структуры:

1) Word – хранит в себе указатель на строку, содержащую слово, длину слова и символ, который отделяет слово от следующего за ним в предложении.
2) Sentence – хранит указатель на массив Word и количество слов.
3) Text – хранит указатель на массив Sentence и количество предложений.

2.2 Взаимодействие с пользователем

Функция main() обеспечивает взаимодействие программы с пользователем и вызывает все остальные функции. Пользователь вводит текст одной строкой (см Ввод). Затем требуется выбрать режим работы программы путём ввода цифры от 1 до 5, которая сохраняется в переменную typeOfProgram. Значения от 1 до 4 вызывают соответствующие функции, которые выполняют одну из подзадач, описанных в условии. При вводе 5 программа закрывается. Пользователь может обработать один и тот же текст несколько раз, пока не закроет программу. Ввести новый текст нельзя.

Перед завершением работы программы происходит очистка памяти с помощью freeText().

2.3 Ввод текста

Пользователь вводит текст, состоящий из предложений. На конце предложений должны стоять точки. Предложения состоят из слов, разделённых пробелом или запятой. Слова – набор латинских букв и цифр.

В структуру Word сохраняется только первый символ разделения, то есть слова в предложении будут разделены одним пробелом или одной запятой.

Перенос строки(\n) является окончанием ввода, то есть в тексте может быть лишь один абзац.

Сразу после ввода текста в нём удаляются повторяющиеся предложения (без учёта регистра букв) с помощью функции deleteDouble().

2.4 Вывод текса

Вывод осуществляется с помощью двух функций:

1) printSentence() – выводит все слова из предложения, если слово не последнее в предложении, то добавляет после него соответствующий разделитель, если последнее – точку.
2) printText() – Для каждого предложения в тексте вызывает printSentence() и разделяет предложения пробелом для большего удобства чтения.

2.5 Первая функция

В функцию sentenceToDeleteChars() передаётся указатель на структуру Text. Внутри для каждого предложения вызывается deleteStartAndEnd(), также эта функция принимает указатель на строку string. Внутри deleteStartAndEnd() вызывается toString(), которая заносить текущее предложение в строку. Затем первый и последний символ строки сравниваются и удаляются, если они равны(без учёта регистра букв). Если строка оказывается пустой, то память, выделенная для string освобождается и возвращается -1. Иначе возвращается 0.

Если deleteStartAndEnd вернула -1, то внутри sentenceToDeleteChars() вызывается deleteSentence() для текущего предложения, иначе очищается память, выделенная под массив Word в предложении с помощью freeSentence(), затем с помощью parseStringToSentence() строка string вносится в Sentence и Word соответственно, заменяя собой удалённое предложение.

2.6 Вторая функция

Пользователь должен ввести номер предложения, в котором будет происходить сортировка, он сохранится в sentenceNum. Если номер меньше 1 или больше количества предложений, то программа выведет «Введен номер

несуществующего предложения» и вернётся к выбору варианта обработки текста.

Иначе вызывается функция wordSort(), она принимает структуру Sentence. Внутри функции вызывается qsort(), в качестве аргументов передаются массив Word, количество слов, размер структуры Word, а также функция wordCmp().

WordCmp() – функция компаратор, она посимвольно сравнивает слова и возвращает: 0 – слова равны, 1 – второе слово должно стоят раньше первого, -1 – первое слово должно стоять перед вторым.

Затем выводится отсортированное предложение.

Разделители закрепляются за словом и выводятся после него, если оно не последнее в предложении, иначе разделитель заменяется точкой.

2.7 Третья функция

В функцию deletePalindrom() передаётся указатель на структуру Text. Для каждого предложения вызывается функция palindromInSentence(). Если она возвращает 1, то предложение удаляется с помощью функции deleteSentence().

palindromInSentence () принимает на структуру Sentence и вызывает isPalindrom(), которая проверят является ли слово палиндромом, для каждого слова в предложении(слово, состоящее из одного символа считается палиндромом). Если для хотя бы одного слова в предложении isPalindrom() вернёт единицу, то palindromInSentence() возвращает 1.

Если после удаления предложений в тексте не осталось ни одного предложения, то программа выведет оповещение об этом, иначе она выведет все предложения, оставшиеся в тексте.

2.8 Четвёртая функция

В функцию findHiddenAgent() передаётся text. flag равна нулю, если в тексте встретится хоть одно слово “HiddenAgent”, то flag становится равным 1. внутри функции для каждого предложения вызывается isAgentInSentence(), которая проверят есть ли в предложении слово “HiddenAgent”. Если

isAgentInSentence() возвращает 1, то в консоль печатается предложение и flag приравнивается к единице.

findHiddenAgent() возвращает значение flag. Если flag равен 0, то программа печатает “ В тексе не встречено ни одного слова “HiddenAgent”.

2.9 Удаление предложений и освобождение памяти

freeSentence(sentence) – вызывает функцию free() для каждого указателя на строку в структуре word, содержащейся в sentence.

freeText(text) – вызывает функцию freeSentece() для каждого предложения внутри text.

deleteSentence(*text, num) – вызывает freeSentence() для предложения с индексом num, затем с помощью memmove() удаляет ссылку на предложение с индексом num и уменьшает значение поля sentenceCount структуры Text на 1.

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены функции стандартной библиотеки языка СИ, разработана и протестирована программа, обрабатывающая текст, согласно условию задания.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Керниган Брайан, Ритчи Деннис — Язык программирования Си. Издательство Вильямс, 2019. 288 с

2. https://stackoverflow.com

3. http://cppstudio.com

ПРИЛОЖЕНИЕ А
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Пример 1: удаление одинаковый символов из начала и конца предложения

```
Введите текст одной строкой:
qwe gds gsdgsdo EwQ. gsdgnoin gsd nogsd.
Повторяющиеся предложения удалены:
qwe gds gsdgsdo EwQ. gsdgnoin gsd nogsd.
Выберите режим работы программы
1 - Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета регистра)
2 - Отсортировать все слова в выбранном предложении в лексикографическом порядке
3 - Удалить все предложения, в которых хотя бы одного слово является палиндромом
4 - Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом
5 - Закрыть программу
1
gds gsdgsdo. gsdgnoin gsd nogsd.
```

Пример 2: сортировка слов в предложении

```
Введите текст одной строкой:
qwe asd zxc, gsdg gs m m.
Повторяющиеся предложения удалены:
qwe asd zxc,gsdg gs m m.
Выберите режим работы программы
1 - Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета регистра)
2 - Отсортировать все слова в выбранном предложении в лексикографическом порядке
3 - Удалить все предложения, в которых хотя бы одного слово является палиндромом
4 - Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом
5 - Закрыть программу
2
Введите номер предложения для сортировки
1
asd gs gsdg m m qwe zxc.
```

Пример 3: удаление предложений с палиндромами

```
Введите текст одной строкой:
qweewq. gmpdsa gdsmgp sdgmdsgmsdp. gmpdsgm d mgpds mgdsgsdmo. gspmdag sdg asa mgpdsgm.
Повторяющиеся предложения удалены:
qweewq. gmpdsa gdsmgp sdgmdsgmsdp. gmpdsgm d mgpds mgdsgsdmo. gspmdag sdg asa mgpdsgm.
Выберите режим работы программы
1 - Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета регистра)
2 - Отсортировать все слова в выбранном предложении в лексикографическом порядке
3 - Удалить все предложения, в которых хотя бы одного слово является палиндромом
4 - Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом
5 - Закрыть программу
3
gmpdsa gdsmgp sdgmdsgmsdp.
```

Пример 4: удаление повторов и вывод предложений с “HiddenAgent”

```
Введите текст одной строкой:
Hiddenagent gdspogsdag. qHiddenAgent gsdagonodsgasd. i am HiddenAgent. Hidden Agent is here. Hiddenagent gdspogsdag. gsdag gdsoigsd.
Повторяющиеся предложения удалены:
Hiddenagent gdspogsdag. qHiddenAgent gsdagonodsgasd. i am HiddenAgent. Hidden Agent is here. gsdag gdsoigsd.
Выберите режим работы программы
1 - Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета
регистра)
2 - Отсортировать все слова в выбранном предложении в лексикографическом порядке
3 - Удалить все предложения, в которых хотя бы одного слово является палиндромом
4 - Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом
5 - Закрыть программу
4
i am HiddenAgent.
```

Пример 5: использование нескольких функций подряд

```
My father is HiddenAgent.


Выберите режим работы программы
1 - Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета
регистра)
2 - Отсортировать все слова в выбранном предложении в лексикографическом порядке
3 - Удалить все предложения, в которых хотя бы одного слово является палиндромом
4 - Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом
5 - Закрыть программу
4
My father is HiddenAgent.


Выберите режим работы программы
1 - Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета
регистра)
2 - Отсортировать все слова в выбранном предложении в лексикографическом порядке
3 - Удалить все предложения, в которых хотя бы одного слово является палиндромом
4 - Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом
5 - Закрыть программу
5
Выход из программы
```

```
Введите текст одной строкой:
i am eight. I am going to school. My father is HiddenAgent.
Повторяющиеся предложения удалены:
i am eight. I am going to school. My father is HiddenAgent.
Выберите режим работы программы
1 - Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета
регистра)
2 - Отсортировать все слова в выбранном предложении в лексикографическом порядке
3 - Удалить все предложения, в которых хотя бы одного слово является палиндромом
4 - Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом
5 - Закрыть программу
2
Введите номер предложения для сортировки
2
am going I school to.

Выберите режим работы программы
1 - Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета
регистра)
2 - Отсортировать все слова в выбранном предложении в лексикографическом порядке
3 - Удалить все предложения, в которых хотя бы одного слово является палиндромом
4 - Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом
5 - Закрыть программу
3
My father is HiddenAgent.
```

Пример 6: введён номер несуществующего предложения для сортировки

```
Введите текст одной строкой:
cxvz gsad tew. gasdp dgmlas etwenm.
Повторяющиеся предложения удалены:
cxvz gsad tew. gasdp dgmlas etwenm.
Выберите режим работы программы
1 - Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета регистра)
2 - Отсортировать все слова в выбранном предложении в лексикографическом порядке
3 - Удалить все предложения, в которых хотя бы одного слово является палиндромом
4 - Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом
5 - Закрыть программу
2
Введите номер предложения для сортировки
3
Введен номер несуществующего предложения
```

Пример 7: в тексте не осталось предложений после удаления палиндромов

```
Введите текст одной строкой:
qweewq gsdo zxv. I am me.
Повторяющиеся предложения удалены:
qweewq gsdo zxv. I am me.
Выберите режим работы программы
1 - Удалить все символы в начале и конце строки так, чтобы в итоге первый и последний символ были различными (без учета регистра)
2 - Отсортировать все слова в выбранном предложении в лексикографическом порядке
3 - Удалить все предложения, в которых хотя бы одного слово является палиндромом
4 - Вывести все предложения в которых есть слово "HiddenAgent" и которое не является первым словом
5 - Закрыть программу
3
В тексте не осталось предложений
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <locale.h>
#define BUF_SIZE 50
#define AGENT "HiddenAgent"

typedef struct Word{
    char *chars;
    size_t len;
    char separator;
} Word;

typedef struct Sentence{
    Word *words;
    size_t wordCount;
} Sentence;

typedef struct Text{
    Sentence *sentence;
    size_t sentenceCount;
} Text;

void getText(Text *text);
int readSentence(Sentence *sentence);
int readWord(Word *word);
void freeSentence(Sentence sentence);
void printText(Text text);
void printSentence(Sentence sentence);
void freeText(Text text);
void deletePalindrom(Text *text);
int palindromInSentence(Sentence sentence);
int findHiddenAgent(Text text);
int isAgentInSentence(Sentence sentence);
void wordSort(Sentence sentence);
char* toString(Sentence sentence);
size_t sentenceLen(Sentence sentence);
void deleteSentence(Text *text, int num);
void deleteDouble(Text *text);
void sentenceToDeleteChars(Text *text);
int deleteStartAndEnd(Sentence *sentence, char **string);
Sentence parseStringToSentence(char **string);

int main() {
    setlocale(LC_ALL, "RUS");
    wprintf(L"Введите текст одной строкой:\n");
    Text text;
    getText(&text);
    deleteDouble(&text);
    wprintf(L"Повторяющиеся предложения удалены:\n");
    printText(text);
    int isHiddenAgentHere;
    int sentenceNum;
```

```
    int typeOfProgram = 0;
    while(typeOfProgram != 5){
        wprintf(L"Выберите режим работы программы\n");
        wprintf(L"1 - Удалить все символы в начале и конце строки так,
чтобы в итоге первый и последний символ были различными (без учета
регистра)\n");
        wprintf(L"2 - Отсортировать все слова в выбранном предложении в
лексикографическом порядке\n");
        wprintf(L"3 - Удалить все предложения, в которых хотя бы одного
слово является палиндромом\n");
        wprintf(L"4 - Вывести все предложения в которых есть слово
“HiddenAgent” и которое не является первым словом\n");
        wprintf(L"5 - Закрыть программу\n");
        scanf("%d", &typeOfProgram);
        switch (typeOfProgram) {
            case 1:
                sentenceToDeleteChars(&text);
                printText(text);
                break;
            case 2:
                wprintf(L"Введите номер предложения для сортировки\n");
                scanf("%d", &sentenceNum);
                if(sentenceNum > text.sentenceCount || sentenceNum < 1){
                    wprintf(L"Введен        номер        несуществующего
предложения\n");
                }
                else{
                    wordSort(text.sentence[sentenceNum - 1]);
                    printSentence(text.sentence[sentenceNum - 1]);
                }
                break;
            case 3:
                deletePalindrom(&text);
                if(text.sentenceCount == 0){
                    wprintf(L"В тексте не осталось предложений");
                }
                else {
                    printText(text);
                }
                break;
            case 4:
                isHiddenAgentHere = findHiddenAgent(text);
                if(isHiddenAgentHere == 0){
                    wprintf(L"В  тексе  не  встречено  ни  одного  слова
“HiddenAgent”");
                }
                break;
            case 5:
                wprintf(L"Выход из программы\n");
                break;
            default:
                wprintf(L"Введено неверное число\n");
                break;
        }
        printf("\n\n");
    }
    freeText(text);
```

```
    return 0;
}

void deleteSentence(Text *text, int num){
    freeSentence(text->sentence[num]);
    memmove(text->sentence + num, text->sentence + num + 1, (text-
>sentenceCount - num - 1) * sizeof(Sentence));
    text->sentenceCount--;
}

void deleteDouble(Text *text){
    for(int i1 = 0; i1 < text->sentenceCount - 1; i1++){
        for(int i2 = i1 + 1; i2 < text->sentenceCount; i2++){
            if(text->sentence[i1].wordCount           !=           text-
>sentence[i2].wordCount){
                continue;
            }
            int flag = 1;
            for(int i = 0; i < text->sentence[i1].wordCount; i++){
                if((strcasecmp(text->sentence[i1].words[i].chars,   text-
>sentence[i2].words[i].chars))  ||  (text->sentence[i1].words[i].separator
!= text->sentence[i2].words[i].separator)){
                    flag = 0;
                    break;
                }
            }
            if(flag){
                deleteSentence(text, i2);
            }

        }
    }
}

int wordCmp(const void *a, const void *b){
    const Word *word1 = (Word*) a;
    const Word *word2 = (Word*) b;
    int i = 0;
    while (tolower(*(word1->chars + i)) == tolower(*(word2->chars + i))){
        if(i == (word1->len - 1) && (word1->len == word2->len)){
            return 0;
        }
        if(i == word1->len - 1){
            return -1;
        }
        if(i == word2->len - 1){
            return 1;
        }
        i++;
    }
    return (tolower(word1->chars[i]) - tolower(word2->chars[i]));
}

void wordSort(Sentence sentence){
    qsort(sentence.words, sentence.wordCount, sizeof(Word), wordCmp);
}
```

```
int findHiddenAgent(Text text){
    int flag = 0;
    for(int i = 0; i < text.sentenceCount; i++){
        if(isAgentInSentence(text.sentence[i])){
            flag = 1;
            printSentence(text.sentence[i]);
            printf("\n");
        }
    }
    return flag;
}

int isAgentInSentence(Sentence sentence){
    for(int i = 0; i < sentence.wordCount; i++){
        if(strcmp(sentence.words[i].chars,AGENT) == 0){
            return 1;
        }
    }
    return 0;
}

int isPalindrom(Word word){
    for(int i = 0; i < word.len / 2 + 1; i++){
        if(!(word.chars[i] == word.chars[word.len - i - 1] || (i ==
word.len - i - 1))) return 0;
    }
    return 1;
}

int palindromInSentence(Sentence sentence){
    for(int i = 0; i < sentence.wordCount; i++){
        if(isPalindrom(sentence.words[i])) {
            return 1;
        }
    }
    return 0;
}

void deletePalindrom(Text *text){
    for(int i = 0; i < text->sentenceCount;){
        if(palindromInSentence(text->sentence[i])){
            deleteSentence(text, i);
        }
        else{
            i++;
        }
    }
}

void getText(Text *text){
    int size = BUF_SIZE;
    int sentenceStatus; // 2 = конец текста, 3 = конец текста, но
последнее предложение введено без точки
    text->sentence = malloc(BUF_SIZE * sizeof(Sentence));
    text->sentenceCount = 0;
    do{
        if(text->sentenceCount == BUF_SIZE - 1){
```

```
            size += BUF_SIZE;
            text->sentence    =    realloc(text->sentence,    size    *
sizeof(Sentence));
        }
        sentenceStatus        =        readSentence(&text->sentence[text-
>sentenceCount]);
        text->sentenceCount++;
    }while(sentenceStatus != 2 && sentenceStatus != 3);
    if(sentenceStatus == 2) text->sentenceCount--;
}

int readSentence(Sentence *sentence){
    int wordStatus; // 0 = конец предложения, 2 = конец текста
    int size = BUF_SIZE;
    sentence->words = malloc(size * sizeof(Word));
    sentence->wordCount = 0;
    do{
        if(sentence->wordCount == BUF_SIZE - 1){
            size += BUF_SIZE;
            sentence->words    =    realloc(sentence->words,    size    *
sizeof(Word));
        }
        wordStatus = readWord(&sentence->words[sentence->wordCount]);
        sentence->wordCount++;
    }while(wordStatus == 1);
    return wordStatus;
}

int readWord(Word *word){
    int result = 1;
    int size = BUF_SIZE;
    int keepGoing = 1;
    word->chars = malloc(size * sizeof(char));
    word->len = 0;
    char ch;

    do{
        ch = (char)getchar();
        if(ch == '\n') return 2;
    }while(ch == ' ');

    do{
        if(word->len == BUF_SIZE - 1){
            size += BUF_SIZE;
            word->chars = realloc(word->chars, size * sizeof(char));
        }
        word->chars[word->len] = ch;
        word->len++;
        ch = (char) getchar();
        if(ch == '.') {
            result = 0;
            keepGoing = 0;
        }
        if(ch == ' '  || ch == ','){
            word->separator = ch;
            keepGoing = 0;
        }
```

```
        if(ch == '\n'){
            keepGoing = 0;
            result = 3;
        }
    }while(keepGoing);
    word->chars[word->len] = '\0';
    return result;
}

void freeText(Text text){
    for(int i = 0; i < text.sentenceCount; i++){
        freeSentence(text.sentence[i]);
    }
}

void freeSentence(Sentence sentence){
    for(int i = 0; i < sentence.wordCount; i++){
        free( sentence.words[i].chars);
    }
}

void printText(Text text){
    for(int i = 0; i < text.sentenceCount; i++){
        printSentence(text.sentence[i]);
        if(i != text.sentenceCount - 1) printf(" ");
    }
    printf("\n");
}

void printSentence(Sentence sentence){
    for(int i = 0; i < sentence.wordCount; i++){
        if(i != sentence.wordCount - 1){
            if(sentence.words)
                printf("%s%c",                  sentence.words[i].chars,
sentence.words[i].separator);
        }
        else{
            printf("%s", sentence.words[i].chars);
        }
    }
    printf(".");
}

size_t sentenceLen(Sentence sentence){
    size_t len = 0;
    for(int i = 0; i < sentence.wordCount; i++){
        len += (sentence.words[i].len + 1);
    }
    return len;
}

char* toString(Sentence sentence){
    size_t len = sentenceLen(sentence);
    size_t index = 0;
    char *string = malloc(sizeof(char) * len);
    for(int i = 0; i < sentence.wordCount; i++){
        strcpy(string + index, sentence.words[i].chars);
```

```
        index += sentence.words[i].len;
        string[index] = sentence.words[i].separator;
        index++;
    }
    string[index - 1] = '\0';
    return string;
}



void sentenceToDeleteChars(Text *text){
    char **string = malloc(sizeof(char*));
    int status;
    for(int i = 0; i < text->sentenceCount;){
        status = deleteStartAndEnd(&(text->sentence[i]), string);
        if(status == -1){
            deleteSentence(text, i);
        }
        else{
            freeSentence(text->sentence[i]);
            text->sentence[i] = parseStringToSentence(string);
            i++;
        }

    }
}

Sentence parseStringToSentence(char **string){
    Sentence newSentence;
    newSentence.wordCount = 0;
    int size = BUF_SIZE;
    newSentence.words = malloc(sizeof(Word) * size);
    size_t len = strlen(*string);
    int j;
    int i = 0;
    while(**(string + i) == ' ' || **(string + i) == ',') i++;
    for(; i < len; i++) {
        if (newSentence.wordCount == BUF_SIZE - 1) {
            size += BUF_SIZE;
            newSentence.words   =   realloc(newSentence.words,   size   *
sizeof(Word));
        }
        newSentence.words[newSentence.wordCount].len = 0;
        int wordSize = BUF_SIZE;
        newSentence.words[newSentence.wordCount].chars = malloc(wordSize
* sizeof(char));
        for (j = 0; *(*string + i + j) != ' ' && *(*string + i + j) !=
',' && *(*string + i + j) != '.' &&
                    *(*string + i + j) != '\0'; j++) {
            if (newSentence.words[newSentence.wordCount].len == size - 2)
{
                wordSize += BUF_SIZE;
                newSentence.words[newSentence.wordCount].chars          =
realloc(newSentence.words[newSentence.wordCount].chars,

wordSize * sizeof(char));
            }
```

```
newSentence.words[newSentence.wordCount].chars[newSentence.words[newSente
nce.wordCount].len] = *(*string +

i + j);
            newSentence.words[newSentence.wordCount].len++;
        }
        if (*(*string + i + j) == ' ' || *(*string + i + j) == ',')
            newSentence.words[newSentence.wordCount].separator          =
*(*string + i + j);

newSentence.words[newSentence.wordCount].chars[newSentence.words[newSente
nce.wordCount].len] = '\0';
        newSentence.wordCount++;
        i += j;
    }
    return newSentence;
}

int deleteStartAndEnd(Sentence *sentence, char **string){
    *string = toString(*sentence);
    size_t len = strlen(*string);
    while((toupper((*string)[0]) == toupper((*string)[len - 1])) && (len
> 0)){
        memmove(*string, *string + sizeof(char), len * sizeof(char));
        len--;
        memmove(*string  +  sizeof(char)  *  (len  -  1),  sentence  +
sizeof(char) * len, 2);
        len--;
        if(len <= 0) {
            free(*string);
            return -1;
        }
    }
    return 0;
}
```