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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка строк на языке Си

Студентка гр. 0382 _____ Деткова А.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студентка Деткова А.С.

Группа 0382

Тема работы: обработка строк на языке Си

Исходные данные:
Программа должна быть написана на языке программирования Си. Обязательно деление программы на функции, каждая из которых отвечает за какое-то действие. Программа получает на вход текст, состоящий из предложений, разделенных запятыми. Предложения состоят из слов, разделенных пробелами или запятыми. Для хранения введенного текста нужно реализовать структуры Sentence и Text. Программа должна быть способна обрабатывать как латинсиские, так и кириллические символы.
Содержание пояснительной записки:
Разделы «Содержание», «Введение», «Заключение», «Список использованных источников», «Цели и задачи», «Отчет о выполненной работе».

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 24.11.2020

Дата сдачи реферата: 28.12.2020

Дата защиты реферата: 29.12.2020

Студентка _____ Деткова А.С.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В ходе курсовой работы была разработана программа на языке Си, которая получает на вход от пользователя текст и обрабатывает его в соответствие с поставленными задачами. Обработка и вывод текста производились при помощи использования функций следующих стандартных заголовочных файлов: stdio.h,stdlib.h, wchar.h, wctype.h, locale.h. Для удобства использования был осуществлен базовый интерфейс, в виде сообщений в терминал, которые упрощают пользование программой. Для сборки программы был составлен Makefile.

СОДЕРЖАНИЕ

1.	Введение	6
2.	Цели и задачи работы	7
3.	Ход выполнения работы	9
3.1	Считывание текста	9
3.2	Решение подзадачи №1	9
3.3	Решение подзадачи №2	10
3.4	Решение подзадачи №3	10
3.5	Решение подзадачи №4	10
3.6	Пункт №5, когда пользователь ничего не выбрал	10
4.	Тестирование	11
5.	Заключение	14
6.	Список использованных источников	15
7.	Приложение А. Исходный код	16

1. ВВЕДЕНИЕ

В курсовой работе осуществляется разработка программы по обработке текста предложений, разделенных запятыми. Пользователю на выбор предлагает решить, какое действие из 5 предложенных ему нужно выполнить. Для этого реализован простейший интерфейс в виде сопроводительных инструкций, выводимых в в поток вывода на консоль пользователя. Для сборки программ используется Makefile. Разработка программы велась на ОС Ubuntu через приложение блокнот и терминал.

2. ЦЕЛИ И ЗАДАЧИ РАБОТЫ

Задание:

Вариант 2

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

Для хранения предложения и для хранения текста требуется реализовать структуры Sentence и Text

Программа должна сохранить (считать) текст в виде динамического массива предложений и оперировать далее только с ним. Функции обработки также должны принимать на вход либо текст (Text), либо предложение (Sentence).

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. Вывести все предложения, в которых каждое слово удовлетворяет введенной строке-условию. Строка условия содержит: символы, * - 0 или больше любых символов, ? – один любой символ. В строке-условия не может быть больше одного *. Например, строка условие может иметь вид “?а?а*н”, которой соответствуют слова “фазан” и “барабан”.
2. Отсортировать предложения по средней длине слов в предложении.
3. Преобразовать предложения так, чтобы слова располагались в порядке уменьшения длины слова.
4. Удалить все предложения, в котором больше 5 или меньше 2 слов.

Все сортировки и операции со строками должны осуществляться с использованием функций стандартной библиотеки. Использование

собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Каждую подзадачу следует вынести в отдельную функцию, функции сгруппировать в несколько файлов (например, функции обработки текста в один, функции ввода/вывода в другой). Также, должен быть написан Makefile.

Цель: освоить считывание текста с клавиатуры, написать программу, которая будет обрабатывать текст в соответствие с поставленной задачей, научиться применять полученные в процессе обучения знания.

Задачи:

1. Написать программу на языке Си
2. Осуществить считывание текста с клавиатуры пользователя
3. Обработать текст в соответствии с поставленной задачей
4. Написать функции обработки текста
5. Сделать сборку программы в Makefile

3.ХОД ВЫПОЛНЕНИЯ РАБОТЫ

3.1 СЧИТЫВАНИЕ ТЕКСТА

Считывание текста происходит в функции *main()*, файл *text.c*. Для хранения текста, а также каждого предложения, реализованы структуры: *Sentence* и *Text*. Объявления структур находятся в файле *structurs.h*. Поля структуры *Text* — *struct Sentence** (указатель на структуру *Sentence*) и *sent_count* (количество предложений в тексте). У структуры *Sentence* поля: *cur_sentence* — само предложение со знаками препинания, пробелами и точкой в конце; *len* — длина предложения; *num_words* — количество слов; ** *words* — массив указателей на слова в предложении; * *dots* — массив знаков препинания в той последовательности, в которой они шли в предложении.

По своей сути весь текст представлен в виде массива структур *Sentence* длиной *sent_count*. Далее вся работа осуществляется только со структурами.

В начале работы программы, на экран выводится приветственное сообщение, которое сообщает пользователю, что окончанием ввода текста является символ переноса строки или *enter*.

После чего текст считывается в динамические массивы строк и символов структур *Sentence* и *Text*.

Последующая обработка программы предполагает выбор со стороны пользователя. Выводится соответствующее сообщение и программа считывает команду, введенную пользователем.

3.2 РЕШЕНИЕ ПОДЗАДАЧИ №1.

Если пользователь введет команду 1, то программа начнет обработку текста. Но сначала будет считана строка-условие. Далее, указатель на текст передается в функцию *search()* заголовочный файл *search.h* файл с определением функции *search.c*. Эта функция принимает на вход указатель на структуру *Text* и указатель на считанную строку-условие. Возвращает новую структуру *Text*, в которой остались только подходящие условию предложения, далее эти предложения выводятся на экран.

3.3 РЕШЕНИЕ ПОДЗАДАЧИ №2.

Решение подзадачи №2 реализуется в функции *sort_len_words(),* которая использует функцию компаратор cmp_2(), заголовочный файл *sort_len_words.h* файла *sort_len_words.* На вход она получает указатель на текст, где обрабатывает его и сортирует предложения текста. После чего отсортированный текст выводится на экран из вызываемой функции.

3.4 РЕШЕНИЕ ПОДЗАДАЧИ №3.

В функцию-сортировщик *less_order()* передаются предложения по очереди там они обрабатываются и выводятся на экран. Также функция *less_order()* пользуется функцией компаратором *cmp().*

3.5 РЕШЕНИЕ ПОДЗАДАЧИ №4

В функцию *del_sent()* передается указатель на текст. После обработки текст измененный текст выводится на экран.

3.6 ЕСЛИ ПОЛЬЗОВАТЕЛЬ НЕ ОПРЕДЕЛИЛСЯ С ВЫБОРОМ.

Текст выводится на экран.

4. ТЕСТИРОВАНИЕ

1. Запуск утилиты *make*:

```
anna@anna-VirtualBox:~/CW$ make
gcc -c text.c
gcc -c del_sent.c
gcc -c less_order.c
gcc -c search.c
gcc -c sort_len_word.c
gcc text.o del_sent.o less_order.o search.o sort_len_word.o -o text
```

2. Начало работы программы:

```
anna@anna-VirtualBox:~/CW$ ./text
Вас приветствует программа, которая обрабатывает текст!
Введите текст, который нужно отредактировать.
(Чтобы закончить ввод, следует нажать enter)
```

Программа выводит приветственное сообщение и предлагает пользователю ввести сообщение

3. Решение подзадачи №1:

```
anna@anna-VirtualBox:~/CW$ ./text
Вас приветствует программа, которая обрабатывает текст!
Введите текст, который нужно отредактировать.
(Чтобы закончить ввод, следует нажать enter)
апелбпа аррвртбла,атбта апродбаа аабаа.аббаа ааааабаа,апрбдб.аа двль,жжлпвн.
Текст считан, выберете дальнейшие действия с программой:
Введите '1', если хотите вывести все предложения, в которых каждое слово удовлет
воряет введенной строке-условию.
Введите '2.', если хотите отсортировать предложения по средней длине слов в пред
ложении.
Введите '3.', если хотите преобразовать предложения так, чтобы слова располагали
сь в порядке уменьшения длины слова.
Введите '4.', если хотите удалить все предложения, в котором больше 5 или меньше
 2 слов.
Введите '5.', если не нужно выполнять никаких действий.
1
а*?б?а
Введите строку-условие, ввод завершится после нажатия клавиши enter
Помните!
Что в сроке-условии может быть один символ * и сколь угодно много символов?.
апелбпа аррвртбла,атбта апродбаа аабаа.anna@anna-VirtualBox:~/CW$
```

4. Решение подзадачи №2:

```
anna@anna-VirtualBox:~/CW$ ./text
Вас приветствует программа, которая обрабатывает текст!
Введите текст, который нужно отредактировать.
(Чтобы закончить ввод, следует нажать enter)
овлвлыл,лвлвлвлвл,вомтмттсовор,вовововоо воововрср,вововововоо.о тстст,ово.ВОтмтм
оо вошшултм.цшшцш с.о р ам тш.
Текст считан, выберете дальнейшие действия с программой:
Введите '1', если хотите вывести все предложения, в которых каждое слово удовле
творяет введенной строке-условию.
Введите '2.', если хотите отсортировать предложения по средней длине слов в пре
дложении.
Введите '3.', если хотите преобразовать предложения так, чтобы слова располагал
ись в порядке уменьшения длины слова.
Введите '4.', если хотите удалить все предложения, в котором больше 5 или меньш
е 2 слов.
Введите '5.', если не нужно выполнять никаких действий.
2.
о р ам тш.о тстст,ово.цшшцш с.ВОтмтмоо вошшултм.овлвлыл,лвлвлвлвл,вомтмттсовор,
вовововоо воововрср,вововововоо.anna@anna-VirtualBox:~/CW$
```

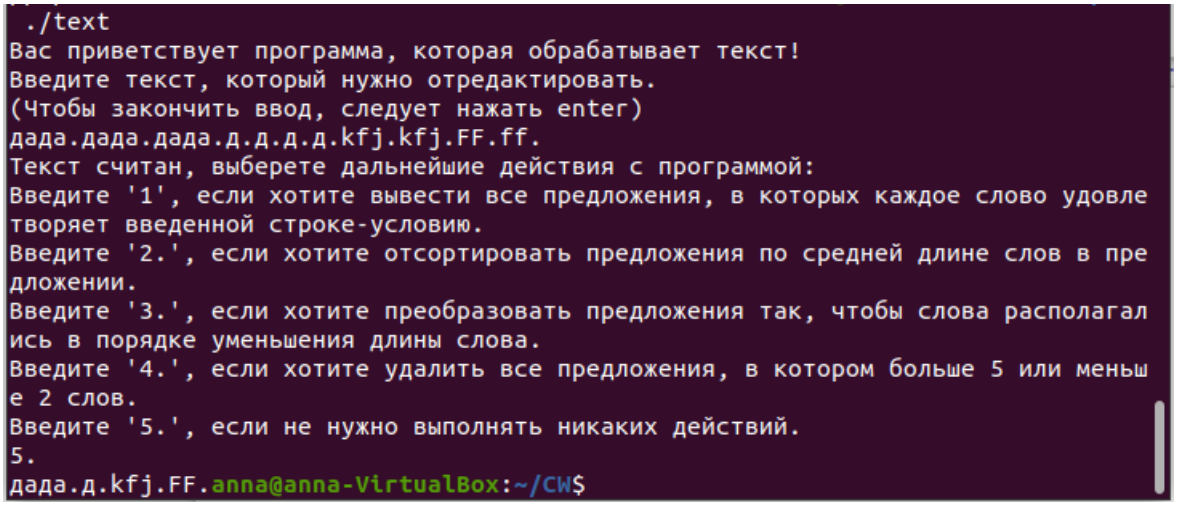
5. Решение подзадачи №3:

```
Вас приветствует программа, которая обрабатывает текст!
Введите текст, который нужно отредактировать.
(Чтобы закончить ввод, следует нажать enter)
оовооовр иси щущуг иисп п.оыоыну щщ аыаыаыаыааы дди.рурри,ов афсс,лалал иии.ПРИ
ВЕТ.привет.
Текст считан, выберете дальнейшие действия с программой:
Введите '1', если хотите вывести все предложения, в которых каждое слово удовле
творяет введенной строке-условию.
Введите '2.', если хотите отсортировать предложения по средней длине слов в пре
дложении.
Введите '3.', если хотите преобразовать предложения так, чтобы слова располагал
ись в порядке уменьшения длины слова.
Введите '4.', если хотите удалить все предложения, в котором больше 5 или меньш
е 2 слов.
Введите '5.', если не нужно выполнять никаких действий.
3.
п иси иисп щущуг оовооовр.щщ дди оыоыну аыаыаыаыааы.ов,иии афсс,рурри лалал.ПРИ
ВЕТ.привет.anna@anna-VirtualBox:~/CW$
```

6. Решение подзадачи №4:

```
Вас приветствует программа, которая обрабатывает текст!
Введите текст, который нужно отредактировать.
(Чтобы закончить ввод, следует нажать enter)
ррвр пвпвпп оовво пвпвп лвлв.овов пвп.воов впвп пвпв.овово,оово шушу рвпвв вдвд
 исис.
Текст считан, выберете дальнейшие действия с программой:
Введите '1', если хотите вывести все предложения, в которых каждое слово удовле
творяет введенной строке-условию.
Введите '2.', если хотите отсортировать предложения по средней длине слов в пре
дложении.
Введите '3.', если хотите преобразовать предложения так, чтобы слова располагал
ись в порядке уменьшения длины слова.
Введите '4.', если хотите удалить все предложения, в котором больше 5 или меньш
е 2 слов.
Введите '5.', если не нужно выполнять никаких действий.
4.
ррвр пвпвпп оовво пвпвп лвлв.овов пвп.воов впвп пвпв.anna@anna-VirtualBox:~/CW$
```

7. Решение подзадачи №5:



5. ЗАКЛЮЧЕНИЕ

Целью работы над курсовой работой было создание стабильно работающей программы по обработке текста. В результате такая программа была получена. Ошибок в ее работе выявлено не было. Для достижения результата были использованы структуры, стандартные библиотеки языка Си, функции сортировки.

6. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. https://www.cplusplus.com/

2. Керниган Б. и Ритчи Д. Язык программирования Си. М.: Вильямс, 1978 288 с

3. ПРОГРАММИРОВАНИЕ Учебно-методическое пособие / сост: К. В. Кринкин, Т. А. Берленко, М. М. Заславский, К. В. Чайка.: СПбГЭТУ «ЛЭТИ», 2018. 34с.

7. ПРИЛОЖЕНИЕ А

ТЕКСТ ПРОГРАММЫ ПРИЛОЖЕНИЯ

Файл: text.c

```
#include <locale.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wchar.h>

#include "structures.h"
#include "del_sent.h"
#include "sort_len_word.h"
#include "search.h"
#include "less_order.h"
#include "cmp.h"
#include "cmp2.h"



int main(){
    setlocale(LC_ALL, "");

    wprintf(L"Вас приветствует программа, которая обрабатывает текст!\n"
             "Введите текст, который нужно отредактировать.\n"
             "(Чтобы закончить ввод, следует нажать enter)\n");

    wchar_t symbol = L'0';
    int flag = 1, txt_buf = 10;
    struct Text text;

    text.sent_count = 0;
    text.sentence = (struct Sentence*)malloc(txt_buf * sizeof(struct
Sentence));

    while(flag == 1){

        symbol = getwchar();

        if (symbol == L'\n'){
            flag = 0;
            break;
        }

        int buffer = 20;
        struct Sentence sentence;
        sentence.cur_sentence = (wchar_t*)malloc(buffer *
sizeof(wchar_t));
        sentence.len = 0;


        int num_letters = 0, wordsize = 20, sentsize = 10;
        sentence.num_words = 0;
        wchar_t* word = malloc(wordsize * sizeof(wchar_t));
```

```
        sentence.words = malloc(sentsize * sizeof(wchar_t*));
        sentence.dots = malloc(sentsize * sizeof(wchar_t));

        while(symbol != L'.'){

            sentence.cur_sentence[sentence.len] = symbol;
            sentence.len ++;

            if ((symbol != L',') && (symbol != L' ')){
                word[num_letters] = symbol;
                num_letters ++;
                if (num_letters + 2 == wordsize){
                    wordsize = wordsize + 10;
                    word = realloc(word, wordsize * sizeof(wchar_t));
                }
            }
            else{
                word[num_letters] = L'\0';

                if (sentence.num_words + 2 == sentsize){
                    sentsize = sentsize + 5;
                    sentence.words = realloc(sentence.words, sentsize *
sizeof(wchar_t*));
                    sentence.dots = realloc(sentence.dots, sentsize *
sizeof(wchar_t));
                }

                sentence.words[sentence.num_words] = word;
                sentence.dots[sentence.num_words] = symbol;
                sentence.num_words ++;

                word = malloc(wordsize * sizeof(wchar_t));
                num_letters = 0;

            }

            if (sentence.len + 2 == buffer){
                buffer = buffer + 20;
                sentence.cur_sentence = realloc(sentence.cur_sentence,
buffer * sizeof(wchar_t));
            }
            symbol = getwchar();
        }

        word[num_letters] = L'\0';

        sentence.words[sentence.num_words] = word;
        sentence.dots[sentence.num_words] = symbol;
        sentence.num_words ++;
        sentence.dots[sentence.num_words] = '\0';

        sentence.cur_sentence[sentence.len] = symbol;
        sentence.len ++;

        sentence.cur_sentence[sentence.len] = '\0';

        int sign = 0;
```

```
        for (int i = 0; i < text.sent_count; i ++)
            if (sentence.len == text.sentence[i].len){
                int k = 0;
                for (int j = 0; j < sentence.len; j ++){
                    if (tolower(sentence.cur_sentence[j]) ==
tolower(text.sentence[i].cur_sentence[j]))
                        k ++;
                }
                if (k == sentence.len)
                    sign = 1;
                }

        if (sign == 0){
            text.sentence[text.sent_count] = sentence;
            text.sent_count ++;
        }

        if (text.sent_count == txt_buf){
            txt_buf = txt_buf + 10;
            text.sentence = realloc(text.sentence, txt_buf *
sizeof(struct Sentence));
        }

    }

    int a;

    wprintf(L"Текст считан, выберете дальнейшие действия с программой:\n"
            "Введите '1', если хотите вывести все предложения, в которых
каждое слово удовлетворяет введенной строке-условию.\n"
            "Введите '2.', если хотите отсортировать предложения по
средней длине слов в предложении.\n"
            "Введите '3.', если хотите преобразовать предложения так,
чтобы слова располагались в порядке уменьшения длины слова.\n"
            "Введите '4.', если хотите удалить все предложения, в котором
больше 5 или меньше 2 слов.\n"
            "Введите '5.', если не нужно выполнять никаких действий.\n");

    wscanf(L"%d\n", &a);

    switch (a){
    case 1:
        wprintf(L"Введите строку-условие, ввод завершится после нажатия
клавиши enter\n"
                "Помните!\n"
                "Что в сроке-условии может быть один символ * и сколь
угодно много символов?.\n");
        struct Text result_text;
        int buf = 10, cur_size = 0;
        wchar_t* str_for_search = malloc(buf * sizeof(wchar_t));
        wchar_t ch = getwchar();
        while (ch != '\n'){
            str_for_search[cur_size] = ch;
            cur_size ++;
            if (cur_size + 2 == buf){
                buf = buf + 5;
                str_for_search = realloc(str_for_search, buf *
sizeof(wchar_t));
```

```
            }
            ch = getwchar();
        }
        str_for_search[cur_size] = '\0';

        result_text = search(&text, &str_for_search);
        for (int i = 0; i < result_text.sent_count; i ++)
            wprintf(L"%ls", result_text.sentence[i].cur_sentence);
        break;
    case 2:
        sort_len_word(&text);
        for (int i = 0; i < text.sent_count; i ++)
            wprintf(L"%ls", text.sentence[i].cur_sentence);
        break;
    case 3:
        for (int i = 0; i < text.sent_count; i ++){
            less_order(&text.sentence[i]);
            wprintf(L"%ls", text.sentence[i].cur_sentence);
        }
        break;
    case 4:
        delete_sentences(&text);
        for (int i = 0; i < text.sent_count; i ++)
            wprintf(L"%ls", text.sentence[i].cur_sentence);
        break;
    case 5:
        for (int i = 0; i < text.sent_count; i ++)
            wprintf(L"%ls", text.sentence[i].cur_sentence);
        break;
    }

    for (int i = 0; i < text.sent_count; i ++){
        for(int j = 0; j < text.sentence[i].num_words; j ++){
            free(text.sentence[i].words[j]);
        }
        free(text.sentence[i].words);
        free(text.sentence[i].dots);
    }
    free(text.sentence);

    return 0;
```

Файл: search.c

```
#include <locale.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wchar.h>
#include "structures.h"

struct Text search(struct Text* txt,  wchar_t ** str){
    struct Text res;
    wchar_t* mask = *str;
    int res_buf = 10;
    res.sentence = malloc(res_buf * sizeof(struct Sentence));
    res.sent_count = 0;
```

```
    for (int i = 0; i < txt->sent_count; i ++){
        int mark = 1;
        for (int k = 0; k < txt->sentence[i].num_words; k ++){
            if (wcslen(txt->sentence[i].words[k]) >= wcslen(mask) - 1){
                int str_indx = 0, msk_indx = 0;
                while ((mask[msk_indx] != '*') && (mark)){
                    if (mask[msk_indx] == '?'){
                        msk_indx ++;
                        str_indx ++;
                        wprintf(L"-*-%d\n", mark);
                    }
                    else if (mask[msk_indx] == txt->sentence[i].words[k]
[str_indx]){
                        msk_indx ++;
                        str_indx ++;
                    }
                    else mark = 0;

                }
                if (mark){
                    msk_indx = wcslen(mask) - 1;
                    str_indx = wcslen(txt->sentence[i].words[k]) - 1;
                }
                while ((mask[msk_indx] != '*') && (mark)){
                    if (mask[msk_indx] == '?'){
                        msk_indx --;
                        str_indx --;
                    }
                    else if (mask[msk_indx] == txt->sentence[i].words[k]
[str_indx]){
                        msk_indx --;
                        str_indx --;
                    }
                    else mark = 0;
                }

            }
            else mark = 0;
        }
        if (mark){
            res.sentence[res.sent_count] = txt->sentence[i];
            res.sent_count ++;
            if (res_buf == res.sent_count + 2){
                res_buf = res_buf + 10;
                res.sentence = realloc(res.sentence, res_buf *
sizeof(struct Sentence));
            }
        }
    }
    return res;
}
```

Файл: search.h

```
struct Text search(struct Text* txt,  wchar_t ** str);
```

Файл: less_order.c

```
#include <locale.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wchar.h>
#include "structures.h"

int cmp(const void* a, const void* b){
    wchar_t** aa = (wchar_t**)a;
    wchar_t** bb = (wchar_t**)b;
    if (wcslen(*aa) > wcslen(*bb))
        return 1;
    if (wcslen(*aa) == wcslen(*bb))
        return 0;
    if (wcslen(*aa) < wcslen(*bb))
        return -1;
}


void less_order(struct Sentence* sent){

    qsort(sent->words, sent->num_words, sizeof(wchar_t*), cmp);
    int j = 0;
    for (int i = 0; i < sent->num_words;i ++){
        for (int k = 0; k < wcslen(sent->words[i]); k ++){
            sent->cur_sentence[j] = sent->words[i][k];
            j ++;
        }
        sent->cur_sentence[j] = sent->dots[i];
        j ++;
    }
}
```

Файл: less_order.h

```
void less_order(struct Sentence* sent);
```

Файл: cmp.h

```
int cmp(const void* a, const void* b);
```

Файл: sort_len_word.c

```
#include <locale.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wchar.h>
#include "structures.h"

int cmp_2(const void * a, const void * b){
    struct Sentence* aa = (struct Sentence*)a;
    struct Sentence* bb = (struct Sentence*)b;
```

```
    int len_aa = 0, len_bb = 0;
    for (int i = 0; i < aa->num_words; i ++)
        len_aa = len_aa + wcslen(aa->words[i]);
    len_aa = len_aa / aa->num_words;
    for (int i = 0; i < bb->num_words; i ++)
        len_bb = len_bb + wcslen(bb->words[i]);
    len_bb = len_bb / bb->num_words;

    if (len_aa > len_bb)
        return 1;
    if (len_aa == len_bb)
        return 0;
    if (len_aa < len_bb)
        return -1;
}

void sort_len_word(struct Text * txt){
    qsort(txt->sentence, txt->sent_count, sizeof(struct Sentence),
cmp_2);
}
```

Файл: sort_len_word.h

```
void sort_len_word(struct Text * txt);
```

Файл: cmp2.h

```
int cmp_2(const void * a, const void * b);
```

Файл: del_sent.c

```
#include <locale.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <wchar.h>
#include "structures.h"

void delete_sentences(struct Text* txt){
    int i = 0;
    while (i < txt->sent_count){
        if ((txt->sentence[i].num_words < 2) || (txt-
>sentence[i].num_words > 5)){
            for (int k = i; k < txt->sent_count - 1; k ++){
                txt->sentence[i] = txt->sentence[i + 1];
            }
            txt->sent_count --;
            i --;
        }
        i ++;
    }
}
```

Файл: del_sent.h

```
void delete_sentences(struct Text* txt);
```

Файл: del_sent.h

```
void delete_sentences(struct Text* txt);
```

Файл: structures.h

```
struct Sentence{
    wchar_t* cur_sentence;
    int len, num_words;
    wchar_t ** words, * dots;
};

struct Text{
    struct Sentence *sentence;
    int sent_count;
};
```

Файл: Makefile

```
all: text
text: text.o del_sent.o less_order.o search.o sort_len_word.o
     gcc text.o del_sent.o less_order.o search.o sort_len_word.o -o text
text.o: text.c
     gcc -c text.c
del_sent.o: del_sent.c
     gcc -c del_sent.c
search.o: search.c
     gcc -c search.c
less_order.o: less_order.c
     gcc -c less_order.c
sort_len_word.o: sort_len_word.c
     gcc -c sort_len_word.c
clean:
     rm *.o text
```