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.

Студент гр. 1382 _____ Коренев Д. А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ.

Студент Коренев Д. А.

Группа 1382

Тема работы: Обработка строк на языке Си.

Исходные данные:

Вариант 5

- 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) Распечатать каждое слово и количество его повторений в тексте.
2) Заменить каждый символ, который не является буквой, на его код.

3) Отсортировать предложения по количеству латинских букв в предложении.
4) Удалить все предложения, которые содержат специальные символы и не содержат заглавные буквы.

- 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.

Предполагаемый объем пояснительной записки:
Не менее 28 страниц.

Дата выдачи задания: 15.10.2021

Дата сдачи реферата: 23.12.2021

Дата защиты реферата: 25.12.2021

Студент гр. 1382_____ Коренев Д. А.
Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

Курсовая работа заключается в реализации программы для обработки текста на языке Си. Для хранения и работы с текстом были использованы структуры и функции стандартных библиотек языка Си.

Сначала программа выводит подсказку о том, что нужно вводить текст, считывает его и удаляет одинаковые, независимо от регистра, предложения. Выводит подсказку, запрашивает у пользователя, что делать дальше, предусмотрена опция выхода из программы, а так же обработка неправильного ввода. Далее программа выполняет функции, которые вызывает пользователь, пока он не введет значение для выхода из программы. При выходе из программы производятся действия по очистке выделенной памяти.

Пример работы программы приведен в приложении А.

Исходный код программы приведен в приложении Б.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы:

Написать программу, выполняющую действия по обработке текста, выбранные пользователем. Реализация программы происходит с помощью использования структур (для хранения текста, предложений и некоторых данных о них), стандартных библиотек (в том числе wchar.h и wctype.h для работы с кириллическими буквами) и выделения динамической памяти. Сборка программы происходит с помощью Makefile и утилиты make.

Для достижения поставленной цели требуется решить следующие задачи: изучить синтаксис языка программирования C, разработать код программы, написать Makefile, собрать проект, протестировать работоспособность программы.

Программа разработана для операционных систем на базе Linux.

Разработка производилась на операционной системе Ubuntu Linux в IDE Clion и редакторе Vim.

ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Создание и объявление структур

Для работы с текстом созданы структуры Sentence и Text:

Переменные структуры Sentence:

wchar_t* sentence_arr – указатель на массив предложения;

int count_char – количество символов в предложение;

int count_lat – количество латинских букв в предложении;

Переменные структуры Text:

Sentence_s *text_arr – указатель на массив предложений;

int count_sentence – количество предложений;

Структуры объявлены в заголовочном файле structs.h

2.2. Считывание текста

Считывание текста осуществляется при помощи функций read_text() и read_sent() находящихся в заголовочном файле read.c.

Рассмотрим функцию read_text():

В качестве аргумента принимает указатель на текст. В цикле while создается новое предложение, а в его массив символов (т.е. само содержимое предложения) вводятся символы при помощи функции read_sent(). Делается проверка, на перенос строки, если он был – считывание текста завершается. Полученное предложение добавляется в массив предложений текста, увеличивается количество предложений на 1. Если память в массиве предложений закончилась, производится ее перевыделение с бОльшим объемом при помощи стандартной функции realloc().

Функция read_sent():

В качестве аргументов принимает указатель на две переменные структуры предложения: count_char, count_lat, а возвращает указатель на массив символов – предложение. Создается массив типа wchar_t* под который выделяется динамическая память. С помощью цикла while посимвольно считывается предложение. Проверяется, будет пользователь вводить предложение или заканчивает, в первом случае цикл продолжается, иначе возвращается массив из одного символа - \n. В массив символов добавляется считаные символ, длина увеличивается на 1, также проверяется, является ли символ буквой латинского алфавита. При нехватке памяти она увеличивается. После ввода пользователем “.” цикл while заканчивается, в переменным предложения – аргументы функции - присваиваются значения, в конце предложения добавляется нуль-терминатор, и возвращает массив символов.

2.3. Функции обработки и изменения текста.

Функция del_equal_sentences() удаляет из текста два одинаковых предложения независимо от регистра букв в нем, на выход которй принимается указатель на текст. При помощи двух циклов for – первый - для оригинального предложения, второй – для предложения, потенциально эквивалентному первому – сравниваются два предложения. Для ускорения работы программы сравниваются только предложения с одинаковым количеством символов, для сравнения предложения использована функция wcscasecmp описанная в библиотеке wchar.h. Если предложения одинаковые, то при помощи функции memmove сдвигаются все предложения после эквивалентного на 1 влево – копия предложения удалилась. Количество предложений уменьшилось поэтому значение count_sentence текста также уменьшается на 1, а для корректной работы с циклами for значение copy так же уменьшается на 1, чтобы в следующей итерации сравнивалось

предложение на этом же индексе, ведь предложения были смещены на 1 «влево».

Функция del_some_sentence принимает в качестве аргумента текст. В цикле for для каждого предложения создаются переменный «флаги»: flag_s_s и flag_u_c для указания есть ли специальный символ в предложении и заглавная буква соответственно. Дальше в цикле for анализируются символы предложения: проверяется, если символ - не буква, но специальный символ, то флаг специального символа – flag_s_s – «поднимается», становится равным 1, если буква – при помощи функции iswupper(), описанной в библиотеке wchar.h, проверяется заглавная она или нет, и если заглавная – флаг заглавной буквы – flag_u_c – «поднимается», становится равным 1. При выходе из цикла проверяется, если флаг специального символа поднят, а флаг заглавной буквы опущен, то предложение удаляется при помощи функции memmove() подобно удалению предложения в функции del_equal_sent(), также выводится на экран удаленное предложение. После завершения другого цикла for для наглядности выводятся все предложения.

Функция symbol_to_code() принимает на вход текст. При помощь двух циклов for (первый для предложений, второй для символов в предложении) программа анализирует: если символ в предложении не является буквой, то в переменную num присваивается код данного символа. Выделяется память для массива символов, в который при помощи функции itoa записываются посимвольно цифры из переменной num. В массив предложения выделяется память, сдвигаются остальные символы после этого, при помощи функции wcsncpy вставляется код символа в предложение. Длина текста увеличивается, переменная symbol, по которой итерирует цикл for так же увеличивается на длину кода и уменьшается на 1 (сам символ был удален).

Функция itoa() принимает на вход число и указатель на массив символов. При помощи цикла do {} while получаем остаток от деления 10

входного числа, прибавляем к нему ASCII код «0» получаем ASCII код остатка от деления на 10 входного числа, которое записывается в массив, число целочисленно делится на 10. Массив чисел инвертируется при помощи функции reverse и возвращается.

Функция reverse() принимает в качестве аргумента указатель на массив слова, при помощи цикла for меняет местами символы равноудалённых от середины слова.

2.4. Функции вывода текста.

Функция print_text() принимает на вход указатель на текст и попредложно при помощи цикла for выводит предложения с помощью функции wprintf().

Функция text_data() принимает на вход указатель на текст и попредложно при помощи цикла for выводит количество латинских букв в предложении а также само предложение при помощи функции wprintf().

Функция err() принимает на вход число и при помощи оператора switch в зависимости от входного значения выводит, какая произошла ошибка.

Функция count_words_to_dict() принимает на вход указатель на текст. Создается массив для указателей на массив – слово, переменная – количество слов. При помощи функции for для каждого предложения создается его копия, а далее при помощи функции wcstok() предлежение делится на слова. Если слово было, то переменная – количество его повторений увеличивается на 1, а если не было, то оно добавляется в словарь, а значение его повторений становится равным 1. Память освобождается, а функция выводит слова и их количество в тексте.

2.5. Функции исполнители.

Функция main.c. В ней с помощью функции setlocale задается локаль, которая будет использоваться текущей программой. Создается текст, выделяется память для хранения указателей на предложения, печатается

подсказка о готовности программы к вводу предложений. Вызываются функции read_text(), del_equal_sentences(), work().

Функция work() принимает на вход указатель на текст, печатает подсказку о возможных дальнейших действиях. При помощи цикла while считываются данные от пользователя – какую функцию вызвать. Далее оператор switch, в зависимости от значения введенного пользователем, вызывается соответствующая функция либо выводится предупреждение, что значение, введенное пользователем, некорректно.

Функция cmp() используется как функция-компаратор для функции qsort для выполнения одной из задач программы. На вход принимает указатели, создает переменные типа Struct и присваивает им входные значения. Сравнивает предложения по количеству латинских буквы и возвращает соответствующие значения.

2.6. Создание Makefile.

Программа бала разделена на файлы:

structs.h – заголовочный файл, в котором содержится объявление структур Text и Struct;

structs.c – файл, в котором содержатся стурктуры Text и Struct;

main.c – файл, в котором содержатся функции-исполнители;

read.c – файл, в котором содержатся функции вывода текста;

read.h – заголовочный файл, в котором содержатся сигнатуры функций вывода текста

change_sent.c – файл, в котором содержатся функции обработки и изменения текста;

change_sent.h - заголовочный файл, в котором содержатся сигнатуры функций обработки и изменения текст;

print.c – файл, в котором содержатся функции вывода текста;

print.h – файл, в котором содержатся сигнатуры функций вывода текста;

Сборка программы осуществляется с помощью Makefile, в котором прописаны все необходимые цели и зависимости, и утилиты make.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была создана программа для обработки текста. Для хранения текста были реализованы структуры Text и Sentence. Программа обрабатывает введённый текст в соответствии с выбором пользователя. Программа может выполнять следующие действия:

1. Распечатать каждое слово и количество его повторений в тексте.
2. Заменить каждый символ, который не является буквой, на его код.
3. Отсортировать предложения по количеству латинских букв в предложении.
4. Удалить все предложения, которые содержат специальные символы и не содержат заглавные буквы.
5. Вывод текста.
6. Выход из программы.

Программа была разбита на файлы, и написан Makefile.

Программа была успешно протестирована на работоспособность.

Примеры тестирования смотреть в приложении А.

Файлы программы смотреть в приложении Б.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОНИКОВ

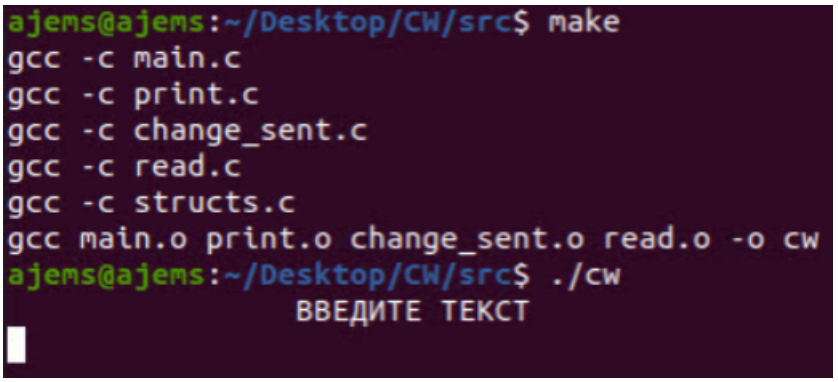
1. Сайт www.c-cpp.ru
2. Сайт https://docs.microsoft.com/ru-ru/cpp/c-runtime-library
3. Сайт https://en.cppreference.com/w/

ПРИЛОЖЕНИЕ А

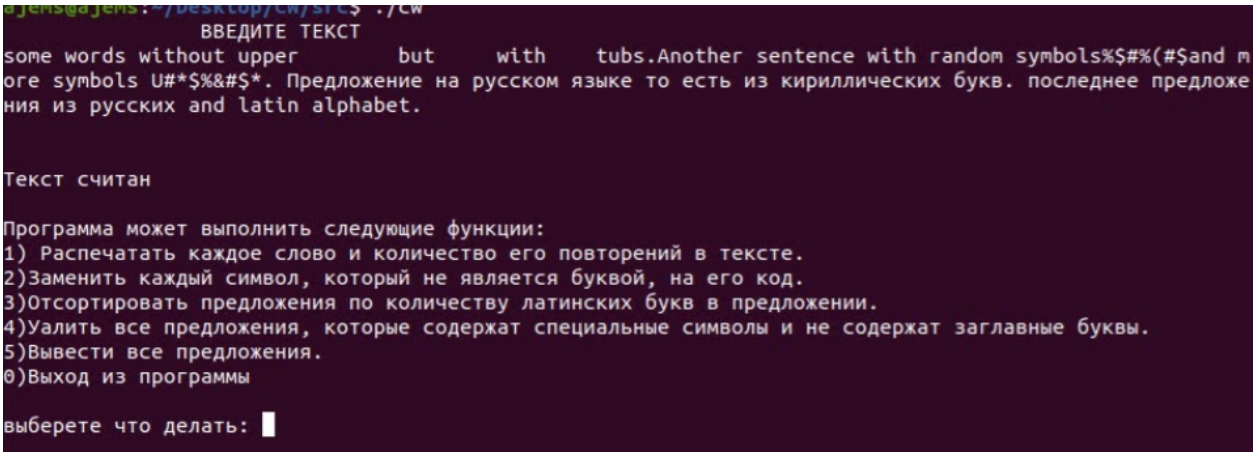
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Тест 1:

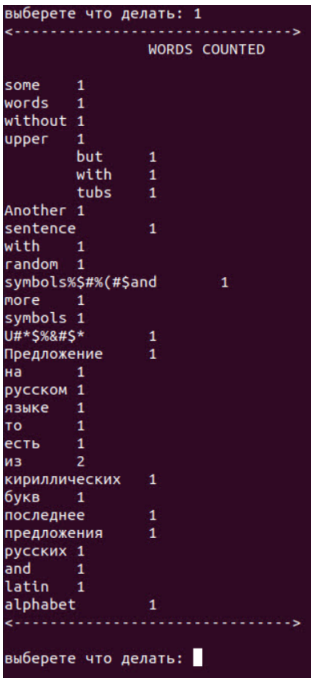
1. Вывод подсказки для пользователя после запуска исполняемого файла:



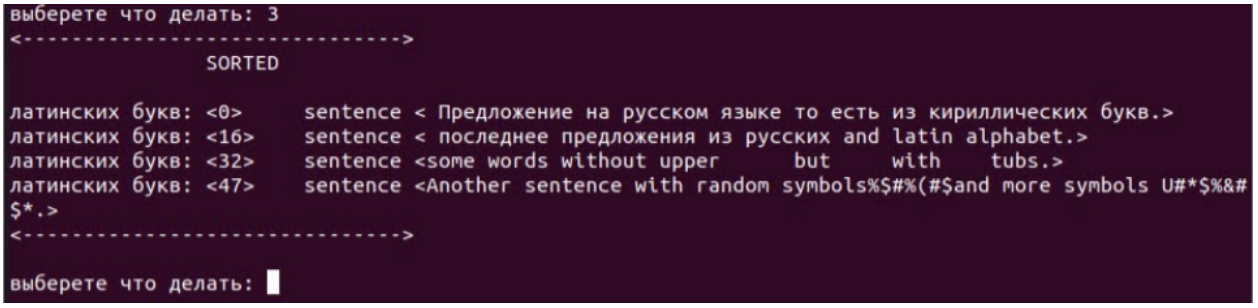
2. Пример введенного текста и вывод меню для пользователя:



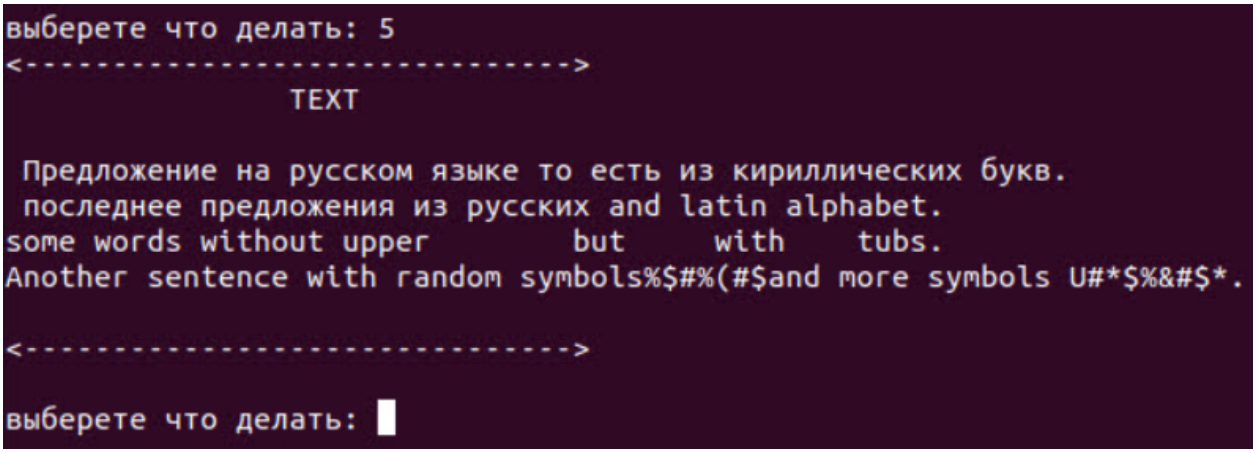
3. Выполнение первой задачи:



4. Выполнение третьей задачи:



5. Выполнение пятой задачи:



6. Выполнение четвертой задачи:

```
выберете что делать: 4
DELETED ---> some words without upper   but     with    tubs.
<------------------------------->
                AFTER DELETE

 Предложение на русском языке то есть из кириллических букв.
 последнее предложения из русских and latin alphabet.
Another sentence with random symbols%$#%(#$and more symbols U#*$%&#$*.
<------------------------------->

выберете что делать:
```

7. Выполнение второй задачи:

```
выберете что делать: 2
<------------------------------>
               TEXT

32Предложение32на32русском32языке32то32есть32из32кириллических32букв46
32последнее32предложения32из32русских32and32latin32alphabet46
Another32sentence32with32random32symbols37363537403536and32more32symbols32U354236373835364246

<------------------------------>

выберете что делать:
```

8. Неверный ввод символа пользователя:

```
выберете что делать: упс
Неверное значение! попробуйте еще раз

выберете что делать:
```

9. Выход:

```
выберете что делать: 0
Программа закончилась
Спасибо за использование!
ajems@ajems:~/Desktop/CW/src$
```

Тест 2:

1. Пример введенного текста и вывод меню для пользователя:

```
ajems@ajems:~/Desktop/CW/src$ ./cw
                ВВЕДИТЕ ТЕКСТ
sagua sgpsuang snagangngang darn adun[adh.adhd h[oiadnia[h bn adgnad фытп эвтпэ фтгрфат
врвю. фрефшв ршфр8р898г*Г%*Н?%*№Н:*(_№;%?("%?№_*:?№;()?;:_№;. о втыпэвот тnzsngp asng a
n       n ngdg . sepupgn gundgdgd h
 sd.hsdh shdh . gsd8y*Y%#*hjt_H n B BSBGDFUBGDUFBGB8DB8ZB baybgnsdhsd. sgdngp uh89eh4t8
9h(UhH&B*YBbf 8yB%$*%)Y#$Y*089teh drgjndfiugndfuignduzh.


Текст считан

Программа может выполнить следующие функции:
1) Распечатать каждое слово и количество его повторений в тексте.
2)Заменить каждый символ, который не является буквой, на его код.
3)Отсортировать предложения по количеству латинских букв в предложении.
4)Уалить все предложения, которые содержат специальные символы и не содержат заглавные
буквы.
5)Вывести все предложения.
0)Выход из программы

выберете что делать:
```

2. Выполнение первой задачи:

```
выберете что делать: 1
<------------------------------->
                WORDS COUNTED

sagua   1
sgpsuang        1
snagangngang    1
darn    1
adun[adh        1
adhd    1
h[oiadnia[h     1
bn      1
adgnad  1
фытп    1
эвтпэ   1
фтгрфатврвю     1
фрефшв  1
ршфр8р898г*Г%*Н?%*№Н    1
*(_№    1
%?("%?№_*       1
?№      1
()?     1
_№      1
о       1
втыпэвот        1
тnzsngp 1
asng    1
an      n       1
ngdg    1
sepupgn 1
gundgdgd        1
h
        1
sd      1
hsdh    1
shdh    1
gsd8y*Y%#*hjt_H 1
n       1
B       1
BSBGDFUBGDUFBGB8DB8ZB   1
baybgnsdhsd     1
sgdngp  1
uh89eh4t89h(UhH&8*YBbf  1
8yB%$*%)Y#$Y*089teh     1
drgjndfiugndfuignduzh   1
<------------------------------->
```

3. Выполнение третьей задачи:

```
выберете что делать: 3
<------------------------------>
               SORTED

латинских букв: <0>     sentence < фрефшв ршфр8р898г*Г%*Н?%*№Н:*(_№;%?("%?№_*:?№;()?;:_
№;.>
латинских букв: <8>     sentence <hsdh shdh .>
латинских букв: <17>    sentence < о втыпзвот тпzsngp asng an   n ngdg .>
латинских букв: <18>    sentence < sepupgn gundgdgd h
 sd.>
латинских букв: <21>    sentence <adhd h[oiadnia[h bn adgnad фытп звтпз фтгрфатврвю.>
латинских букв: <36>    sentence <sagua sgpsuang snagangngang darn adun[adh.>
латинских букв: <41>    sentence < gsd8y*Y%#*hjt_H n B BSBGDFUBGDUFBGB8DB8ZB baybgnsdhs
d.>
латинских букв: <48>    sentence < sgdngp uh89eh4t89h(UhH&B*YBbf 8yB%$*%)Y#$Y*089teh dr
gjndfiugndfuignduzh.>
<------------------------------>
```

4. Выполнение четвёртой задачи:

```
выберете что делать: 4
DELETED --->  о втыпзвот тпzsngp asng an        n ngdg .
DELETED --->  sepupgn gundgdgd h
 sd.
<------------------------------>
               AFTER DELETE

 фрефшв ршфр8р898г*Г%*Н?%*№Н:*(_№;%?("%?№_*:?№;()?;:_№;.
hsdh shdh .
adhd h[oiadnia[h bn adgnad фытп звтпз фтгрфатврвю.
sagua sgpsuang snagangngang darn adun[adh.
 gsd8y*Y%#*hjt_H n B BSBGDFUBGDUFBGB8DB8ZB baybgnsdhsd.
 sgdngp uh89eh4t89h(UhH&B*YBbf 8yB%$*%)Y#$Y*089teh drgjndfiugndfuignduzh.
<------------------------------>
```

5. Выполнение второй задачи:

```
выберете что делать: 2
<------------------------------>
               TEXT

32фрефшв32ршфр56р565756г42Г3742Н6337428470Н58424095847059376340343763847095425863847059
40416359589584705946
hsdh32shdh3246
adhd32h91oiadnia91h32bn32adgnad32фытп32звтпз32фтгрфатврвю46
sagua32sgpsuang32snagangngang32darn32adun91adh46
32gsd56y42Y373542hjt95H32n32B32BSBGDFUBGDUFBGB56DB56ZB32baybgnsdhsd46
32sgdngp32uh5657eh52t5657h40UhH38B42YBbf3256yB3736423741Y3536Y42485657teh32drgjndfiugnd
fuignduzh46

<------------------------------>
```

6. Выполнение пятой задачи:

```
выберете что делать: 5
<------------------------------>
               TEXT

32фрефшв32ршфр56р565756г42Г3742Н6337428470Н58424095847059376340343763847095425863847059
40416359589584705946
hsdh32shdh3246
adhd32h91oiadnia91h32bn32adgnad32фытп32звтпз32фтгрфатврвю46
sagua32sgpsuang32snagangngang32darn32adun91adh46
32gsd56y42Y373542hjt95H32n32B32BSBGDFUBGDUFBGB56DB56ZB32baybgnsdhsd46
32sgdngp32uh5657eh52t5657h40UhH38B42YBbf3256yB3736423741Y3536Y42485657teh32drgjndfiugnd
fuignduzh46

<------------------------------>
```

7. Выход:

```
выберете что делать: 0
Программа закончилась
Спасибо за использование!
ajems@ajems:~/Desktop/CW/src$
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл main.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>
#include "structs.h"
#include "print.h"
#include "change_sent.h"
#include "read.h"
#define STEP 10
#define SIZE 50



int cmp(const void *a, const void *b){
    Sentence_s sent_1 = *(Sentence_s*) a;
    Sentence_s sent_2 = *(Sentence_s*) b;
    if (sent_1.count_lat > sent_2.count_lat) return 1;
    if (sent_1.count_lat < sent_2.count_lat) return -1;
    return 0;
}


void work(Text_s* text){

    wprintf(L"\n\nТекст считан\n\n");
    wprintf(L"Программа может выполнить следующие функции:\n1) Распечатать
каждое слово и количество его повторений в тексте.\n2)Заменить каждый
символ, который не является буквой, на его код.\n3)Отсортировать
предложения по количеству латинских букв в предложении.\n4)Уалить все
предложения, которые содержат специальные символы и не содержат заглавные
буквы.\n5)Вывести все предложения.\n0)Выход из программы\n");

    wchar_t way[SIZE];
    way[0] = L'5';
    while (way[0] != L'0'){
        wprintf(L"\nвыберете что делать: ");
        fgetws(way, SIZE, stdin);

        if (way[2] != L'\0'){
            err(2);
            continue;
        }


        switch ((long int)way[0]){
            case L'1':
                count_words_to_dict(text);
                break;
```

```
            case L'2':
                symbol_to_code(text);
                print_text(text);
                break;
            case L'3':
                qsort(text->text_arr, text->count_sentence,
sizeof(Sentence_s), cmp);
                text_data(text);
                break;
            case L'4':
                del_some_sentence(text);
                break;
            case L'5':
                print_text(text);
                break;
            case L'0':
                wprintf(L"Программа закончилась\nСпасибо за
использование!\n");

                for (int i = 0; i <= text->count_sentence; i++){
                    free(text->text_arr[i].sentence_arr);
                }

                free(text->text_arr);
                break;
            default:
                err(2);
                break;
        }
    }
}


int main(){
    setlocale(LC_ALL, "");
    Text_s text;
    text.count_sentence = 0;
    text.text_arr = malloc(SIZE*sizeof(Sentence_s));

    if (text.text_arr == NULL){
        err(1);
    }

    wprintf(L"\t\tВВЕДИТЕ ТЕКСТ\n");
    read_text(&text);
    del_equal_sentences(&text);
    work(&text);
    return 0;
}
```

Файл print.c

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

```
#include "structs.h"
#define STEP 10
#define SIZE 50

void err(int a){
    switch (a) {
        case 1:
            wprintf(L"Ошибка! Выделение памяти невозможно!\n");
            break;
        case 2:
            wprintf(L"Неверное значение! попробуйте еще раз\n");
            break;
        default:
            break;
    }
}

void print_text(Text_s* text){
    wprintf(L"<------------------------------>\n");
    wprintf(L"\t\tTEXT\n\n");
    for (int i = 0; i < text->count_sentence; i++) {
        wprintf(L"%ls\n", text->text_arr[i].sentence_arr,
text->text_arr[i].count_char);
    }
    wprintf(L"\n<------------------------------>\n");
}

void text_data(Text_s* text){
    wprintf(L"<------------------------------>\n\t\tSORTED\n\n");
    for(int i = 0; i < text->count_sentence; i++){
        wprintf(L"латинских букв: <%ld>\tsentence <%ls>\n",
text->text_arr[i].count_lat, text->text_arr[i].sentence_arr);
    }
    wprintf(L"<------------------------------>\n");
}

void count_words_to_dict(Text_s* text){
    //массив указателей на слова

    wchar_t** dict = (wchar_t**)malloc((SIZE)*sizeof(wchar_t*));

    if (dict == NULL){
        err(1);
    }


    int mem_size_dict = SIZE;
    int* count = (int*)calloc(SIZE, sizeof(int));
    int total_words = 0;


    //для каждого предлоежния
    for (int sent = 0; sent < text->count_sentence; sent++){

        //создаем копию предложения
```

```
        wchar_t* sent_copy =
malloc((wcslen(text->text_arr[sent].sentence_arr)+1) * sizeof (wchar_t));
        if(sent_copy == NULL){
            err(1);
        }

        wcscpy(sent_copy, text->text_arr[sent].sentence_arr);
        wchar_t *pt;
        wchar_t* words = wcstok(sent_copy, L" ,.;:\n\t", &pt);


        while (words != NULL){

            unsigned long len_word = wcslen(words);

            if (words[len_word-1] == L','){
                words[len_word-1] = L'\0';
            }

            int flag = 0;
            //пробегаемся по словам в словаре
            for (int i = 0; i < total_words; i++){

                //нашлось такое слово в словаре
                if (wcscmp(dict[i], words) == 0) {

                    //увеличиить на 1 его кол-во встречаний(?)
                    count[i]++;
                    flag = 1;
                    break;
                }
            }

            //если слово не нашлось
            if (flag == 0){

                if (total_words == mem_size_dict){
                    mem_size_dict += STEP;
                    dict = (wchar_t**)realloc(dict,
(mem_size_dict)*sizeof(wchar_t*));
                    if(dict == NULL){
                        err(1);

                    }

                    //выделитьдоп память для count
                    count = (int*)realloc(count,
(mem_size_dict)*sizeof(int));
                    if(count == NULL){
                        err(1);
                    }
                }

                //создание области памяти для новго слова и указатель
положить в словарь +1 для \0
```

```
                wchar_t* word_to_dict = malloc((wcslen(words)+1)*sizeof
(wchar_t));

                if(word_to_dict == NULL){
                    err(1);
                }

                wcscpy(word_to_dict, words);

                //добавить его в массив и сделать кол-во встречаний = 1,
на 1 слово стало больше (total_words)
                dict[total_words] = word_to_dict;

                count[total_words]++;
                total_words++;
            }

            //получить новое слово
            words = wcstok(NULL, L" ,.;:", &pt);
        }

        //предложение закончилось, память можно освободить
        free(sent_copy);

    }
    wprintf(L"<------------------------------>\n");
    wprintf(L"\t\tWORDS COUNTED\n\n");
    for (int i = 0; i < total_words; i++) {
        wprintf(L"%ls\t%d\n", dict[i], count[i]);
    }
    for (int i = 0; i < total_words; i++) {
        free(dict[i]);
    }
    free(count);
    free(dict);
    wprintf(L"<------------------------------>\n");
}
```

Файл print.h

```
void err(int a);
void count_words_to_dict(Text_s* text);
void print_text(Text_s* text);
void text_data(Text_s* text);
```

Файл change_sent.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>
#include "structs.h"
#include "print.h"
#define STEP 10
#define SIZE 50
```

```
void del_equal_sentences(Text_s* text){
    getchar();

    for (int origin = 0; origin < text->count_sentence; origin++){
        for (int copy = origin+1; copy < text->count_sentence; copy++){

            // ускорения программы, предложения будут сравниваться только
при одинаковом количестве символов
            if (text->text_arr[origin].count_char ==
text->text_arr[copy].count_char) {

                if (wcscasecmp(text->text_arr[origin].sentence_arr,
text->text_arr[copy].sentence_arr) == 0){
                    // удалить copy смещением массива и, соответсвенно,
удалением copy во всем text

                    memmove((text->text_arr)+copy,
(text->text_arr)+copy+1,
((text->count_sentence)-copy)*sizeof(Sentence_s));

                    //стало меньше предложенй
                    text->count_sentence --;
                    //т.к. предложение удалено, курсор остается на том же
индексе
                    copy --;

                }
            }
        }
    }
}

void del_some_sentence(Text_s* text){
    for(int sent =0; sent < text->count_sentence; sent++){
        int flag_s_s = 0;
        int flag_u_c = 0;



        for (int symbol = 0; symbol < text->text_arr[sent].count_char;
symbol++){
            if (iswalpha(text->text_arr[sent].sentence_arr[symbol]) == 0){

                //ЕСЛИ НЕ БУКВЫ
                if ((7 <= text->text_arr[sent].sentence_arr[symbol]) &&
(text->text_arr[sent].sentence_arr[symbol] <= 13)){

                    flag_s_s = 1;
                }
            }
            else{
                if (iswupper(text->text_arr[sent].sentence_arr[symbol])){

                    //флаг для заглавных букв поднят
                    flag_u_c = 1;
                }
```

```
            }
        }


        if (flag_s_s == 1 && flag_u_c == 0){
            wprintf(L"DELETED ---> %ls\n",
text->text_arr[sent].sentence_arr);
            memmove((text->text_arr)+sent, (text->text_arr)+sent+1,
((text->count_sentence)-sent)*sizeof(Sentence_s));
            text->count_sentence--;
            sent--;
        }
    }
    wprintf(L"<------------------------------>\n");
    wprintf(L"\t\tAFTER DELETE\n\n");
    for (int i = 0; i < text->count_sentence; i++) {
        wprintf(L"%ls\n", text->text_arr[i].sentence_arr);
    }
    wprintf(L"<------------------------------>\n");
}

void reverse(wchar_t *s)
{
    unsigned long i, j;
    wchar_t c;

    for (i = 0, j = wcslen(s)-1; i<j; i++, j--) {
        c = s[i];
        s[i] = s[j];
        s[j] = c;
    }
}


int itoa(long int n, wchar_t *s)
{
    int i;
    i = 0;

    do {
        s[i++] = n % 10 + L'0';
    } while ((n /= 10) > 0);

    s[i] = '\0';
    reverse(s);

    return i;
}


void symbol_to_code (Text_s* text){
    for (int sent = 0; sent < text->count_sentence; sent++){
        for(int symbol = 0; symbol < text->text_arr[sent].count_char;
symbol++){
            if (iswalpha(text->text_arr[sent].sentence_arr[symbol]) == 0){
```

```
                //получили код символа
                int num = (int)text->text_arr[sent].sentence_arr[symbol];
                wchar_t* str_num = (wchar_t*)malloc(6 * sizeof(wchar_t));

                if(str_num == NULL){
                    err(1);
                }

                int len = itoa(num, str_num);


                //выделяем память, сдвигаем символы, вставляем слово,
увеличиваем длину предложения
                text->text_arr[sent].sentence_arr =
(wchar_t*)realloc(text->text_arr[sent].sentence_arr,
(text->text_arr[sent].count_char+len)*sizeof(wchar_t));
                if(text->text_arr[sent].sentence_arr == NULL){
                    err(1);
                }

                memmove(text->text_arr[sent].sentence_arr+symbol+len-1,
text->text_arr[sent].sentence_arr+symbol,
(text->text_arr[sent].count_char-symbol+1)*sizeof(wchar_t));

                wcsncpy(text->text_arr[sent].sentence_arr + symbol,
str_num, len);

                text->text_arr[sent].count_char+=len-1;
                symbol += len - 1;

            }
        }
    }
}
```

Файл change_sent.h

```
void symbol_to_code (Text_s* text);
void del_some_sentence(Text_s* text);
void del_equal_sentences(Text_s* text);
```

Файл read.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>
#include "structs.h"
#include "print.h"
#define STEP 10
#define SIZE 50



//чтение предложения
wchar_t* read_sent(int *count_char, int *count_lat){
    int len = 0, lat = 0, size = SIZE;
```

```
    unsigned int c;
    wchar_t* text = malloc(size*sizeof(wchar_t));

    if(text == NULL){
        err(1);
    }

    while(1){
        c = getwchar();

        if (c == L'\n' && len == 0){
            text[0] = c;
            *count_char = 1;
            return text;
        }


        //непосредсвенно считывание в массив
        text[len] = c;

        //латинские
        if (((97 <= c) && (c<= 122)) || ((65 <= c) && (c <= 90))){
            lat++;
        }

        len++;
        if (len == size){
            size += STEP;
            text = (wchar_t*) realloc(text, size*sizeof (wchar_t));

            if(text == NULL){
                err(1);
            }
        }

        if (c == '.'){
            break;
        }
    }

    *count_char = len;
    *count_lat = lat;
    text[len] = '\0';
    return text;
}


void read_text (Text_s* text){

    int real_size_text_arr = SIZE;

    while (1){

        Sentence_s sentence;
```

```
        sentence.sentence_arr = read_sent(&sentence.count_char,
&sentence.count_lat);

        if (sentence.sentence_arr[0] == L'\n'){
            break;
        }

        //добавили в массив предложений нашу структуру предложения
        text->text_arr[text->count_sentence] = sentence;
        text->count_sentence ++;

        //расширение памяти для хранения структур предложений
        if (text->count_sentence == real_size_text_arr){
            text->text_arr = realloc(text->text_arr,
real_size_text_arr+STEP);
            if(text->text_arr == NULL){
                err(1);
            }

            real_size_text_arr += STEP;
        }
    }
}
```

Файл read.h

```
wchar_t* read_sent(int *count_char, int *count_lat);
void read_text (Text_s* text);
```

Файл structs.c

```
#include <wchar.h>


typedef struct Sentence{
    wchar_t* sentence_arr;
    int count_char;
    int count_lat;
}Sentence_s;


typedef struct Text{
    Sentence_s *text_arr;
    int count_sentence;
}Text_s;
```

Файл structs.h

```
#include <wchar.h>

typedef struct Sentence{
    wchar_t* sentence_arr;
    int count_char;
    int count_lat;
}Sentence_s;


typedef struct Text{
    Sentence_s *text_arr;
    int count_sentence;
```

```
}Text_s;
```

Файл Makefile

```
all: main.o print.o change_sent.o read.o structs.o
	gcc main.o print.o change_sent.o read.o -o cw

main.o: main.c print.h change_sent.h read.h structs.h
	gcc -c main.c

print.o: print.c structs.h
	gcc -c print.c

change_sent.o: change_sent.c print.h structs.h
	gcc -c change_sent.c

read.o: read.c structs.h
	gcc -c read.c

structs.o: structs.c structs.h
	gcc -c structs.c

clean:
	rm  *.o
```