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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Редактирование текста

Студент гр. 9383 _____ Корсунов А. А.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2019

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Корсунов А. А.

Группа 9383

Тема работы: Работа с текстом

Исходные данные:

На вход программе подается текст. Программа должна считать текст и обработать его по желанию пользователя.

Содержание пояснительной записки:

«Содержание», «Введение», «Задание курсовой работы», «Файлы программы», «Функции программы», «Примеры работы программы», «Заключение», «Приложение А (только в электронном виде)».

Предполагаемый объем пояснительной записки:

Не менее 16 страниц.

Дата выдачи задания: 14.10.2019

Дата сдачи реферата: 14.12.2019

Дата защиты реферата: 14.12.2019

Студент _____ Корсунов А. А.

Преподаватель _____ Жангиров Т. Р.

АННОТАЦИЯ

В ходе подготовки курсовой работы была написана программа, получающая на вход текст. Текст считывается до нажатия пользователем клавиши <enter> (предполагается, что каждое предложение отделено от другого точкой, а после последней точки в тексте ничего не должно стоять, кроме символа перевода строки). После считывания программа удаляет из текста одинаковые предложения и выводит справку с возможными действиями. Была учтена ситуация, при которой пользователь удалит все предложения.

SUMMARY

During the preparation of the course work, a program was written that receives text for input. The text is read before the user presses the < enter> key (it is assumed that each sentence is separated from the other by a dot, and after the last dot there should be nothing in the text except a newline character). After reading, the program removes the same sentences from the text and displays help with possible actions. A situation was considered in which the user would delete all offers.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: написать программу, которая принимает на вход текст и редактирует его согласно выбору пользоваетеля.

Для реализации данной цели необходимо решить следующие задачи:

1. Изучение работы со стандартной библиотекой СИ.
2. Изучение работы с динамической памятью.
3. Изучение работы с широкими символами, типом wcahr_t, и библиотек wchar.h, wctype.h.
4. Изучение работы с файлом Makefile.
5. Изучение работы со структурами,.
6. Разработка функций для работы с текстом.
7. Группировка функций программы в файлы по назначению, создание заголовочных файлов.
8. Написание файла Makefile.
9. Тестирование программы.
10. Отладка программы.
11. Очистка памяти.

1. ЗАДАНИЕ КУРСОВОЙ РАБОТЫ

Вариант: 12

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. Сделать транслитерацию всех кириллических символов в тексте. Например, подстрока «Какой nice пень» должна принять вид «Kakoy nice pen» (использовать ГОСТ 7.79-2000)
2. Для каждого предложения вывести все специальные символы в порядке уменьшения их кода.
3. Заменить все цифры в тексте их двоичным кодом.
4. Удалить все предложения в которых есть нечетные цифры.

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.

2.ФАЙЛЫ ПРОГРАММЫ

2.1. Файл menu.c

В данном файле вызываются все необходимые функции, которые пользователь выберет, также происходит сам вывод справки с подсказками о выборе возможных вариаций действий.

2.2. Файл struct.h

В данном файле находятся все необходимые библиотеки, а также создаются необходимые структуры .

Структура Text имеет 3 поля: целочисленная переменная senten для хранения количества предложений в тексте, целочисленная переменная size для хранения выделенной памяти и массив типа Sentence.

2.3 Файлы input.c, input.h

В файле input.h объявлена функция файла inp_out.c.

В файле input.c реализована функция, считывающая текст.

2.4 Файлы hatred_repeated.c, hatred_repeated.h

В файле hatred_repeated.h объявлена функция файла hatred_repeated.c.

В файле hatred_repeated.c реализована функция удаления повторяющихся предложений.

2.5 Файлы delete_sentence.c, delete_sentence.h

В файле delete_sentence.h объявлены функции файла delete_sentence.c.

В файле delete_sentence.c реализована функция удаления всех одинаковых предложений, а также две вспомогательные функции.

2.6 Файлы hated_Cyrilli.c, hated_Cyrilli.h

В файле hated_Cyrilli.h объявлена функция файла hated_Cyrilli.c.

В файле hated_Cyrilli.c реализована функция, которая производит транслитерацию кириллических символов.

2.7 special_characters.c, special_characters.h

В файле special_characters.h объявлена функция файла special_characters.c.

В файле special_characters.c реализована функция, которая выводит специальные символы для каждого предложения в порядке уменьшения их кода.

2.8 output.c, output.h

В файле output.h объявлена функция файла output.c.

В файле output.c реализована функция, которая выводит текст в stdout.

2.9 Makefile

В данном файле производится компиляция всей программы.

3. ФУНКЦИИ ПРОГРАММЫ

3.1. input()

В данной функции происходит выделение памяти под вводимый текст, а также создаются переменные, которые помогут сохранить текст в выделенной памяти. В цикле while производится проверка на перевыделение памяти и само ее выделение (если проверка прошла успешна). Также производится определение, куда нужно записать введенный символ и само его записывание. Цикл while будет работать пока после символа точки не встретит символ перевода строки.

3.2 hatred_repeated()

В данной функции создаются перменные (индексы), которые помогут в проверке на определение повторяющихся предложений. В цикле while с помощью функций wcslen() и wcscasecmp(), а также проверке на одно и то же предложении, происходит проверка на одинаковые предложения. Если предложение проходит проверку, то происходит «перетаскивание» предложений влево на один уменьшение text->senten на 1. Каждое новое предложение не проверяется с предыдущими (второй индекс всегда берет начальным значением индекс первого+1).

3.3 hated_Cyrilli()

Данная функция не будет работать, если text->senten = 0, потому как это бессмысленно (осуществляется проверка с помощью if). Если проверка пройдена, то создаются переменные, которые помогут в нахождении и транслитерации кириллических букв. Если проверяемый символ не является кириллическим символом, то ничего не происходит, если является то происходит определение, каким именно кириллическим символом он является, следом происходит замена аналогом Латиницы (если аналог имеет более 2 и

более символов, то происходит перевыделение памяти для предложения). Как только индекс рассматремого предложения станет равен text->senten, произойдет выход из цикла и завершение функции.

3.4 special_characters(), comp()

Функция special_characters(): данная функция не будет работать (но вывод пустого предложения произойдет), если text->senten = 0, потому как это бессмысленно (осуществляется проверка с помощью if). Если проверка пройдена, создаются переменные, которые помогут в нахождении специальных символов. Происходит выделение памяти для двумерного массива, в котором будут храниться предложения специальных символов. Если проверяемый символ не является специальным, то функция ничего с созданным массивом не делает, если является, то происходит запись символа в созданный массив. После этого с помощью функции qsort() и компаратора comp() происходит сортировка символов по уменьшению их кода в каждом предложении. Затем происходит требуемый по заданию вывод и очистка выделенной памяти.

Функция comp(): проверяет, чей код из двух элементов больше и возвращает положительное, отрицательное число или ноль (то, что требует qsort()).

3.5 hatred_odd()

В данной функции вначале создаются переменные, которые помогут в нахождении нечетных цифр. Если проверяемый символ не является символом нечетной цифры, то функция ничего не делает, если является то функция в цикле for производит «перетаскивание предложений» влево и уменьшение text->senten на 1. Цикл будет продолжаться пока индекс рассматриваемого предложения не станет равен text->senten (здесь же производится проверка text->senten на ноль, при прохождении которой выведется соответствующая подсказка).

3.6 output()

Данная функция производит вывод текст в stdout с помощью for.

3.7 main()

В данной функции производится вызов всех необходимых функций, вывод в stdout подсказку о возможных действиях, а также очистка оставшейся памяти.

4. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

4.1. Считывание и вывода текста

Входные данные: *В заключение она запела Casta diva: все восторги, молнией несущиеся мысли в голове, трепет, как иглы, пробегающий по телу, – все это уничтожило Обломова: он изнемог.*

Выходные данные: *В заключение она запела Casta diva: все восторги, молнией несущиеся мысли в голове, трепет, как иглы, пробегающий по телу, – все это уничтожило Обломова: он изнемог.*

4.2. Удаления одинаковых предложений

Входные данные: *Молчи, Соня, я совсем не смеюсь, я ведь и сам знаю, что меня черт тащил. Молчи, Соня, молчи. МОЛЧИ, Соня, молчи. МоЛчи, соня, мОлчИ. Молчи, Соня, я совсем не смеюсь, я ведь и сам знаю, что меня черт тащил.*

Выходные данные: *Молчи, Соня, я совсем не смеюсь, я ведь и сам знаю, что меня черт тащил. Молчи, Соня, молчи.*

Программа удалила из текста первое, второе и последнее одинаковые предложения.

4.3 Транслитерация всех кириллических букв

Входные данные: *Я говорю: отчего люди не летают так, как птицы. Знаешь, мне иногда кажется, что я птица. Когда стоишь на горе, так тебя и тянет лететь. Вот так бы разбежалась, подняла руки и полетела. Попробовать нешто теперь.*

Выходные данные: *YA govoryu: otchego lyudi ne letayut tak, kak pticy`. Znaesh`, mne inogda kazhetsya, chto ya ptica. Kogda stoish` na gore, tak tebya i tyanet letet`. Vot tak by` razbezhalas`, podnyala ruki i poletela. Poprobovat` neshto teper`.*

4.4 Специальные символы в порядке уменьшения их кода

Входные данные: *Дал№%ьше дело :?%?за тобой Росиою. Кто я, №":*? дырокоп Симон, я пробурил тонель;(*?%, но чтобы вести по нему людей№"%*"№:<><><?? я не гожусь.*

Выходные данные:

Предложение 1: % % : ? ? №

*Предложение 2: " " " % % (* * * : : ; < < < > > ? ? ? ? № № №*

4.5 Замена всех цифр в тексте их двоичным кодом

Входные данные: *Зло — это зло. Меньшее, бо́л1ьшее, среднее — всё едино, пропорции условны, а границы размыты. Я не свят154ой от4637шельник, не только одно231 добро тв3орил в жизни. Но если приходитс547я 21выбирать между од6ним злом и другим, я 85предпочитаю не выбирать вообще.*

Выходные данные: *Зло — это зло. Меньшее, бо́л1ьшее, среднее — всё едино, пропорции условны, а границы размыты. Я не свят1101100ой от1001101111шельник, не только одно10111 добро тв11орил в жизни. Но если приходитс101100111я 101выбирать между од110ним злом и другим, я 1000101предпочитаю не выбирать вообще.*

4.6 Удаление всех предложений, в которых присутствуют нечетные цифры.

Входные данные: *I've searched for meaning amidst doubt I've finally figured that part ou1t. And all th66e stories inside me. Feels like I'm bursting at the seam3s. And7 you're here after all.*

Выходные данные: *And all th66e stories inside me.*

ЗАКЛЮЧЕНИЕ

Были изучены библиотеки СИ. Реализованы функции обработки вводимого пользователем текста. Расширены знания о стандартной библиотеке СИ, работе с динамической памятью, работе со структурами. Была написана требуемая по заданию программа и разделена на несколько файлов для более удобной работы с ней. Был написан Makefile для удобной и быстрой компиляции программы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сервис вопросов и ответов по программированию [электронный ресурс]. - URL: https://fooobar.com, (дата обращения: 05.12.2019).

2. Основы программирования на языках Си и C++[электронный ресурс]. - URL: http://cppstudio.com, (дата обращения: 05.12.2019).

3. Онлайн справочник программиста на Си и C++[электронный ресурс]. - URL: http://www.c-cpp.ru, (дата обращения: 05.12.2019).

4. Справочник на Си[электронный ресурс]. - URL: http://www.cplusplus.com , (дата обращения: 05.12.2019).

ПРИЛОЖЕНИЕ А

РАЗРАБОТАННЫЙ КОД

1. Файл sruct.h:

```
#pragma once

#include "struct.h"
#include <stdlib.h>
#include <stdio.h>
#include <wchar.h>
#include <wctype.h>
#include <locale.h>

struct Sentence{
   wchar_t* arr;
};

struct Text{
   int senten;
   int size;
   struct Sentence *text;
};
```

2. Файл menu.c:

```
#include "struct.h"
#include "input.h"
#include "hatred_repeated.h"
#include "hatred_odd.h"
#include "replacement.h"
#include "special_characters.h"
#include "comp.h"
#include "hated_Cyrilli.h"
#include "output.h"

int main()
{
   setlocale(LC_ALL,"");
   struct Text text;
   wchar_t key = ' ';
   wprintf(L"Ожидается ввод текста\n");
   input(&text);
   hatred_repeated(&text);
   while(key != L'0'){
      wprintf(L"Выберите одну из предложенных функций\n");
      wprintf(L"1 - транслетирация всех кириллических букв\n");
      wprintf(L"2 - специальные символы в порядке уменьшения их кода\n");
      wprintf(L"3 - замена всех цифр в тексте их двоичным кодом\n");
         wprintf(L"4 - удаление всех предложений, в которых присутсвуют нечетные
цифры\n");
      wprintf(L"5 - вывести текст\n");
      wprintf(L"Если хотите выйти - введите 0\n");
      key = getwchar();
      getwchar();
      switch(key){
```

```
            case L'0':
                break;
            case L'1':
                hated_Cyrilli(&text);
                break;
            case L'2':
                special_characters(&text);
                wprintf(L"\n");
                break;
            case L'3':
                replacement(&text);
                break;
            case L'4':
                hatred_odd(&text);
                break;
            case L'5':
                output(&text);
                break;
            default:
                wprintf(L"Были введены некорректные данные\n");
                wprintf(L"\n");
                break;
        }
    }
    for(int i = 0; i<text.size; i++){
        free(text.text[i].arr);
    }
    free(text.text);
    return 0;
}
```

3. Файл comp.c:

```
#include "struct.h"
#include "comp.h"

int comp(const void* a, const void* b){
    return *(int*)a - *(int*)b;
}
```

4. Файл comp.h:

```
#pragma once

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

5. Файл hated_Cyrilli.c:

```
#include "struct.h"
#include "hated_Cyrilli.h"

void hated_Cyrilli(struct Text *text){
    if(text->senten != 0){
    int count = 0;
    int index = 0;
    int check = 0;
```

```
                                                    wchar_t*        rus        =
L"ёйцукенгшщзхъфывапролджэячсмитьбюЁЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТ
ЬБЮ";
   while(1){
      if(text->text[count].arr[index] == L'\0'){
         count++;
         index = 0;
         if(count == (text->senten)){
            break;
         }
         else{
            continue;
         }
      }
      for(int i = 0; i < 66; i++){
         if(rus[i] == text->text[count].arr[index]){
            check = 1;
         }
      }
      if(check == 0){
         index++;
         continue;
      }
      else{
         check = 0;
         if(text->text[count].arr[index] == L'a'){
            text->text[count].arr[index] = L'a';
            index++;
            continue;
         }
         if(text->text[count].arr[index] == L'A'){
            text->text[count].arr[index] = L'A';
            index++;
            continue;
         }
         if(text->text[count].arr[index] == L'б'){
            text->text[count].arr[index] = L'b';
            index++;
            continue;
         }
         if(text->text[count].arr[index] == L'Б'){
            text->text[count].arr[index] = L'B';
            index++;
            continue;
         }
         if(text->text[count].arr[index] == L'в'){
            text->text[count].arr[index] = L'v';
            index++;
            continue;
         }
         if(text->text[count].arr[index] == L'В'){
            text->text[count].arr[index] = L'V';
            index++;
            continue;
         }
         if(text->text[count].arr[index] == L'г'){
            text->text[count].arr[index] = L'g';
            index++;
```

```
            continue;
        }
        if(text->text[count].arr[index] == L'Г'){
            text->text[count].arr[index] = L'G';
            index++;
            continue;
        }
        if(text->text[count].arr[index] == L'д'){
            text->text[count].arr[index] = L'd';
            index++;
            continue;
        }
        if(text->text[count].arr[index] == L'Д'){
            text->text[count].arr[index] = L'D';
            index++;
            continue;
        }
        if(text->text[count].arr[index] == L'е'){
            text->text[count].arr[index] = L'e';
            index++;
            continue;
        }
        if(text->text[count].arr[index] == L'Е'){
            text->text[count].arr[index] = L'E';
            index++;
            continue;
        }
        if(text->text[count].arr[index] == L'ё'){
                text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
                text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'y';
            text->text[count].arr[index+1] = L'o';
            index += 2;
            continue;
        }
        if(text->text[count].arr[index] == L'Ё'){
                text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
                text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'Y';
            text->text[count].arr[index+1] = L'O';
            index += 2;
            continue;
        }
        if(text->text[count].arr[index] == L'ж'){
                text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
                text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'z';
            text->text[count].arr[index+1] = L'h';
            index += 2;
```

```
            continue;
        }
        if(text->text[count].arr[index] == L'Ж'){
             text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'Z';
            text->text[count].arr[index+1] = L'H';
            index += 2;
            continue;
        }
        if(text->text[count].arr[index] == L'з'){
            text->text[count].arr[index] = L'z';
            index ++;
            continue;
        }
        if(text->text[count].arr[index] == L'З'){
            text->text[count].arr[index] = L'Z';
            index ++;
            continue;
        }
        if(text->text[count].arr[index] == L'и'){
            text->text[count].arr[index] = L'i';
            index ++;
            continue;
        }
        if(text->text[count].arr[index] == L'И'){
            text->text[count].arr[index] = L'I';
            index ++;
            continue;
        }
        if(text->text[count].arr[index] == L'й'){
            text->text[count].arr[index] = L'j';
            index ++;
            continue;
        }
        if(text->text[count].arr[index] == L'Й'){
            text->text[count].arr[index] = L'J';
            index ++;
            continue;
        }
        if(text->text[count].arr[index] == L'к'){
            text->text[count].arr[index] = L'k';
            index ++;
            continue;
        }
        if(text->text[count].arr[index] == L'К'){
            text->text[count].arr[index] = L'K';
            index ++;
            continue;
        }
        if(text->text[count].arr[index] == L'л'){
            text->text[count].arr[index] = L'l';
            index ++;
            continue;
        }
```

```
if(text->text[count].arr[index] == L'Л'){
   text->text[count].arr[index] = L'L';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'м'){
   text->text[count].arr[index] = L'm';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'М'){
   text->text[count].arr[index] = L'M';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'н'){
   text->text[count].arr[index] = L'n';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'Н'){
   text->text[count].arr[index] = L'N';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'о'){
   text->text[count].arr[index] = L'o';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'О'){
   text->text[count].arr[index] = L'O';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'п'){
   text->text[count].arr[index] = L'p';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'П'){
   text->text[count].arr[index] = L'P';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'р'){
   text->text[count].arr[index] = L'r';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'Р'){
   text->text[count].arr[index] = L'R';
   index ++;
   continue;
}
if(text->text[count].arr[index] == L'с'){
   text->text[count].arr[index] = L's';
   index ++;
```

```
    continue;
}
if(text->text[count].arr[index] == L'C'){
    text->text[count].arr[index] = L'S';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'т'){
    text->text[count].arr[index] = L't';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'Т'){
    text->text[count].arr[index] = L'T';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'у'){
    text->text[count].arr[index] = L'u';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'У'){
    text->text[count].arr[index] = L'U';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'ф'){
    text->text[count].arr[index] = L'f';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'Ф'){
    text->text[count].arr[index] = L'F';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'х'){
    text->text[count].arr[index] = L'x';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'Х'){
    text->text[count].arr[index] = L'X';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'ц'){
    text->text[count].arr[index] = L'c';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'Ц'){
    text->text[count].arr[index] = L'C';
    index ++;
    continue;
}
if(text->text[count].arr[index] == L'ч'){
```

```
                text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text->text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'c';
            text->text[count].arr[index+1] = L'h';
            index += 2;
            continue;
        }
        if(text->text[count].arr[index] == L'Ч'){
                text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text->text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'C';
            text->text[count].arr[index+1] = L'H';
            index += 2;
            continue;
        }
        if(text->text[count].arr[index] == L'ш'){
                text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text->text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L's';
            text->text[count].arr[index+1] = L'h';
            index += 2;
            continue;
        }
        if(text->text[count].arr[index] == L'Ш'){
                text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text->text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'S';
            text->text[count].arr[index+1] = L'H';
            index += 2;
            continue;
        }
        if(text->text[count].arr[index] == L'щ'){
                text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text->text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 2; i > index + 2; i--){
               text->text[count].arr[i] = text->text[count].arr[i-2];
            }
            text->text[count].arr[index] = L's';
            text->text[count].arr[index+1] = L'h';
            text->text[count].arr[index+2] = L'h';
            index += 3;
            continue;
        }
        if(text->text[count].arr[index] == L'Щ'){
                text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text->text[count].arr) + 1 + 4) * sizeof(wchar_t));
```

```
            for(int i = wcslen(text->text[count].arr) + 2; i > index + 2; i--){
               text->text[count].arr[i] = text->text[count].arr[i-2];
            }
            text->text[count].arr[index] = L'S';
            text->text[count].arr[index+1] = L'H';
            text->text[count].arr[index+2] = L'H';
            index += 3;
            continue;
         }
         if(text->text[count].arr[index] == L'ъ'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'`';
            text->text[count].arr[index+1] = L'`';
            index += 2;
            continue;
         }
         if(text->text[count].arr[index] == L'Ъ'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'`';
            text->text[count].arr[index+1] = L'`';
            index += 2;
            continue;
         }
         if(text->text[count].arr[index] == L'ы'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'y';
            text->text[count].arr[index+1] = L'`';
            index += 2;
            continue;
         }
         if(text->text[count].arr[index] == L'Ы'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'Y';
            text->text[count].arr[index+1] = L'`';
            index += 2;
            continue;
         }
         if(text->text[count].arr[index] == L'ь'){
            text->text[count].arr[index] = L'`';
            index ++;
            continue;
         }
```

```
        if(text->text[count].arr[index] == L'ь'){
           text->text[count].arr[index] = L'`';
           index ++;
           continue;
        }
        if(text->text[count].arr[index] == L'э'){
              text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
           for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
              text->text[count].arr[i] = text->text[count].arr[i-1];
           }
           text->text[count].arr[index] = L'e';
           text->text[count].arr[index+1] = L'`';
           index += 2;
           continue;
        }
        if(text->text[count].arr[index] == L'Э'){
              text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
           for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
              text->text[count].arr[i] = text->text[count].arr[i-1];
           }
           text->text[count].arr[index] = L'E';
           text->text[count].arr[index+1] = L'`';
           index += 2;
           continue;
        }
        if(text->text[count].arr[index] == L'ю'){
              text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
           for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
              text->text[count].arr[i] = text->text[count].arr[i-1];
           }
           text->text[count].arr[index] = L'y';
           text->text[count].arr[index+1] = L'u';
           index += 2;
           continue;
        }
        if(text->text[count].arr[index] == L'Ю'){
              text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
           for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
              text->text[count].arr[i] = text->text[count].arr[i-1];
           }
           text->text[count].arr[index] = L'Y';
           text->text[count].arr[index+1] = L'U';
           index += 2;
           continue;
        }
        if(text->text[count].arr[index] == L'я'){
              text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
           for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
              text->text[count].arr[i] = text->text[count].arr[i-1];
           }
           text->text[count].arr[index] = L'y';
           text->text[count].arr[index+1] = L'a';
           index += 2;
```

```
            continue;
        }
        if(text->text[count].arr[index] == L'Я'){
              text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'Y';
            text->text[count].arr[index+1] = L'A';
            index += 2;
            continue;
        }
     }
  }}
}
```

6. Файл hated_Cyrilli.h:

```
#pragma once

void hated_Cyrilli(struct Text *text);
```

7. Файл hatred_odd.c:

```
#include "struct.h"
#include "hatred_odd.h"

void hatred_odd(struct Text *text){
   int count = 0;
   int index= 0;
   while(1){
      if(text->text[count].arr[index] == L'\0'){
         count++;
         index = 0;
         if(count == (text->senten)){
            if(text->senten == 0){
                     wprintf(L"Были удалены все предложения, так как другим функция
нужно для обработки хотя бы одно преложение, вам следует выйти из программы и
ввести новый текст.\n");
            }
            break;
         }
         else{
            continue;
         }
      }
       if((text->text[count].arr[index] != L'7') && (text->text[count].arr[index] != L'1') &&
(text->text[count].arr[index] != L'3') && (text->text[count].arr[index] != L'5') && (text-
>text[count].arr[index] != L'9')){
         index++;
         continue;
      }
      else{
         for(int i = count; i < (text->senten); i++){
            if(wcslen(text->text[i].arr) < wcslen(text->text[i+1].arr)){
```

```
                        text->text[i].arr = (wchar_t*)realloc(text->text[i].arr, (wcslen(text->text[i+1].arr)+1) * sizeof(wchar_t));
            }
            wcscpy(text->text[i].arr, text->text[i+1].arr);
         }
         text->senten = text->senten - 1;
         index = 0;
         if(count == text->senten){
            if(text->senten == 0){
                  wprintf(L"Были удалены все предложения, так как другим функция нужно для обработки хотя бы одно преложение, вам следует выйти из программы и ввести новый текст.\n");
            }
            break;
         }
      }
   }
}
```

8. Файл hatred_odd.h:

```
#pragma once

void hatred_odd(struct Text *text);
```

9. Файл hatred_repeated.c:

```
#include "struct.h"
#include "hatred_repeated.h"

void hatred_repeated(struct Text *text){
   int indexI = 0;
   int indexJ = 0;
   while(1){
            if((indexI != indexJ) && (wcslen(text->text[indexI].arr) == wcslen(text->text[indexJ].arr)) && (!(wcscasecmp(text->text[indexI].arr, text->text[indexJ].arr)))){
         for(int k = indexJ; k < (text->senten); k++){
            if(wcslen(text->text[k].arr) < wcslen(text->text[k+1].arr)){
                        text->text[k].arr = (wchar_t*)realloc(text->text[k].arr, (wcslen(text->text[k+1].arr)+1) * sizeof(wchar_t));
            }
            wcscpy(text->text[k].arr, text->text[k+1].arr);
         }
         text->senten -= 1;
         indexJ -= 1;
      }
      if((indexJ == (text->senten) - 1) && (indexI != (text->senten) - 1)){
         indexI++;
         indexJ = indexI + 1;
      }
      else{
         indexJ++;
      }
      if(indexI == (text->senten) - 1){
         break;
      }
   }
```

```
}
```

10. Файл hatred_repeated.h:

```
#pragma once

void hatred_repeated(struct Text *text);
```

11.Файл input.c:

```
#include "struct.h"
#include "input.h"

void input(struct Text *text){
   text->senten = 5;
   text->size = 5;
   int count = 0;
   wchar_t letter;
   int index = 0;
   int check = 1;
   text->text = (struct Sentence*)calloc((text->size), sizeof(struct Sentence));
   for(int i = 0; i < 5; i++){
      text->text[i].arr = (wchar_t*)calloc(30, sizeof(wchar_t));
   }
   while(check == 1){
      letter = getwchar();
      if((index == 0) && (letter == L'\n')){
            check = 0;
                continue;
      }
      if((index == 0) && (letter != L'\n')){
         if(((count+1) % 5 == 0) && (count != 0)){
                 text->text = (struct Sentence*)realloc(text->text, (count+6) * sizeof(struct
Sentence));
            text->size += 5;
            for(int i = count+1; i < count+6; i++){
               text->text[i].arr = (wchar_t*)calloc(30, sizeof(wchar_t));
            }
         }
         text->text[count].arr[index] = letter;
         index++;
         continue;
      }
      if((index != 0) && (letter != L'.')){
         if((index+1) % 30 == 0){
                 text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (index+30) *
sizeof(wchar_t));
         }
         text->text[count].arr[index] = letter;
         index++;
         continue;
      }
      if((index != 0) && (letter == L'.')){
         if((index+1) % 30 == 0){
                 text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (index+2) *
sizeof(wchar_t));
         }
```

```
            text->text[count].arr[index] = letter;
            text->text[count].arr[index+1] = L'\0';
            if(getwchar() == L'\n'){
                    break;
              }
            index = 0;
            count++;
            continue;
        }
    }
    text->senten = count+1;
}
```

12. Файл input.h

```
#pragma once

void input(struct Text *text);
```

13. Файл output.c

```
#include "struct.h"
#include "output.h"

void output(struct Text *text){
    for(int i = 0; i<text->senten; i++){
                wprintf(L"%ls ", text->text[i].arr);
             }
    wprintf(L"\n");
}
```

14. Файл output.h

```
#pragma once

void output(struct Text *text);
```

15. Файл replacement.c

```
#include "struct.h"
#include "replacement.h"

void replacement(struct Text *text){
    wchar_t* numbers = L"23456789";
    int check = 0;
    int count = 0;
    int index = 0;
    if(text->senten != 0){
    while(1){
        if(text->text[count].arr[index] == L'\0'){
            count++;
            index = 0;
            if(count == (text->senten)){
                break;
            }
            else{
                continue;
            }
        }
        for(int i = 0; i < 8; i++){
            if(text->text[count].arr[index] == numbers[i]){
```

```
            check = 1;
         }
      }
      if(check == 0){
         index++;
         continue;
      }
      else{
         check = 0;
         if(text->text[count].arr[index] == L'2'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'1';
            text->text[count].arr[index+1] = L'0';
            index += 2;
            continue;
         }
         if(text->text[count].arr[index] == L'3'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 1; i > index + 1; i--){
               text->text[count].arr[i] = text->text[count].arr[i-1];
            }
            text->text[count].arr[index] = L'1';
            text->text[count].arr[index+1] = L'1';
            index += 2;
            continue;
         }
         if(text->text[count].arr[index] == L'4'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 2; i > index + 2; i--){
               text->text[count].arr[i] = text->text[count].arr[i-2];
            }
            text->text[count].arr[index] = L'1';
            text->text[count].arr[index+1] = L'0';
            text->text[count].arr[index+2] = L'0';
            index += 3;
            continue;
         }
         if(text->text[count].arr[index] == L'5'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 2; i > index + 2; i--){
              text->text[count].arr[i] = text->text[count].arr[i-2];
            }
            text->text[count].arr[index] = L'1';
            text->text[count].arr[index+1] = L'0';
            text->text[count].arr[index+2] = L'1';
            index += 3;
            continue;
         }
         if(text->text[count].arr[index] == L'6'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
```

```
            for(int i = wcslen(text->text[count].arr) + 2; i > index + 2; i--){
               text->text[count].arr[i] = text->text[count].arr[i-2];
            }
            text->text[count].arr[index] = L'1';
            text->text[count].arr[index+1] = L'1';
            text->text[count].arr[index+2] = L'0';
            index += 3;
            continue;
         }
         if(text->text[count].arr[index] == L'7'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 2; i > index + 2; i--){
               text->text[count].arr[i] = text->text[count].arr[i-2];
            }
            text->text[count].arr[index] = L'1';
            text->text[count].arr[index+1] = L'1';
            text->text[count].arr[index+2] = L'1';
            index += 3;
            continue;
         }
         if(text->text[count].arr[index] == L'8'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 3; i > index + 3; i--){
               text->text[count].arr[i] = text->text[count].arr[i-3];
            }
            text->text[count].arr[index] = L'1';
            text->text[count].arr[index+1] = L'0';
            text->text[count].arr[index+2] = L'0';
            text->text[count].arr[index+3] = L'0';
            index += 4;
            continue;
         }
         if(text->text[count].arr[index] == L'9'){
               text->text[count].arr = (wchar_t*)realloc(text->text[count].arr, (wcslen(text-
>text[count].arr) + 1 + 4) * sizeof(wchar_t));
            for(int i = wcslen(text->text[count].arr) + 3; i > index + 3; i--){
               text->text[count].arr[i] = text->text[count].arr[i-3];
            }
            text->text[count].arr[index] = L'1';
            text->text[count].arr[index+1] = L'0';
            text->text[count].arr[index+2] = L'0';
            text->text[count].arr[index+3] = L'1';
            index += 4;
            continue;
         }
      }
   }}
}
```

16. Файл replacement.h

```
#pragma once

void replacement(struct Text *text);
```

17. Файл special_characters.c

```
#include "struct.h"
#include "special_characters.h"
#include "comp.h"

void special_characters(struct Text *text){
   if(text->senten != 0){
   wchar_t* characters = L"~`!@\"№#;$\%^:?&*()-_+=|\\/\'><";
   int check = 0;
   int count = 0;
   int index_characters = 0;
   int index = 0;
   wchar_t** arr_characters = (wchar_t**)calloc((text->senten), sizeof(wchar_t*));
   for(int i = 0; i < (text->senten); i++){
                  arr_characters[i] = (wchar_t*)calloc(wcslen(text->text[i].arr) + 1,
sizeof(wchar_t));
   }
   while(1){
      if(text->text[count].arr[index] == L'\0'){
         arr_characters[count][index_characters] = L'\0';
         count++;
         index = 0;
         index_characters = 0;
         if(count == (text->senten)){
            break;
         }
         else{
            continue;
         }
      }
      for(int i = 0; i < 27; i++){
         if(text->text[count].arr[index] == characters[i]){
            check = 1;
         }
      }
      if(check == 0){
         index++;
         continue;
      }
      else{
         check = 0;
         arr_characters[count][index_characters] = text->text[count].arr[index];
         index_characters++;
         index++;
      }
```

```
    }
    for(int i = 0; i<(text->senten); i++){
        qsort(arr_characters[i], wcslen(arr_characters[i]), sizeof(wchar_t), comp);
    }
    for(int i = 0; i<(text->senten); i++){
        wprintf(L"Предложение %d: ", i+1);
        for(int j = 0; j<(wcslen(arr_characters[i])); j++){
            wprintf(L"%lc ", arr_characters[i][j]);
        }
        wprintf(L"\n");
    }
    for(int i = 0; i<(text->senten); i++){
        free(arr_characters[i]);
    }
    free(arr_characters);
    }
}
```

18. Файл special_characters.h

```
#pragma once

void special_characters(struct Text *text);
```

19. Файл Makefile

```
CC=gcc
all: menu

menu: menu.o input.o hatred_repeated.o hatred_odd.o replacement.o special_characters.o comp.o hated_Cyrilli.o output.o
	$(CC) menu.o input.o hatred_repeated.o hatred_odd.o replacement.o special_characters.o comp.o hated_Cyrilli.o output.o -o menu

menu.o: menu.c

	$(CC) -c menu.c

input.o: input.c
	$(CC) -c input.c

hatred_repeated.o: hatred_repeated.c
	$(CC) -c hatred_repeated.c

replacement.o: replacement.c
```

```
	$(CC) -c replacement.c

special_characters.o: special_characters.c
	$(CC) -c special_characters.c

comp.o: comp.c
	$(CC) -c comp.c

hated_Cyrilli.o: hated_Cyrilli.c
	$(CC) -c hated_Cyrilli.c

output.o: output.c
	$(CC) -c output.c


clean:
	rm -rf *.o
```