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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Работа с текстом

Студент гр. 9383 _____ Чумак М.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2019

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Чумак М.А.

Группа 9383

Тема работы: Работа с текстом

Исходные данные:

Язык программирования Си, сборка программы под Linux

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 19 страниц.

Дата выдачи задания: 14.10.2019

Дата сдачи реферата: 21.12.2019

Дата защиты реферата: 21.12.2019

Студент _____ Чумак М.А.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

Была разработана программа, которая получает текст и редактирует его по пожеланию пользователя. Текст считывается до нажатия кнопки «enter». После считывания текста программа удаляет одинаковые предложения и ниже выводит диалоговое меню, описывающее действия от той или иной нажатой клавиши в диапазоне от «1» до «4». Для выхода из программы используется клавиша «0». Для вывода текущего состояния текста необходимо ввести число «5». Во всех остальных случаях программа выведет подсказку о неправильно введённом значении и попросит пользователя ввести значение снова.

SUMMARY

A program was developed that receives and edit the text at the request of the user. The text is read before pressing the enter button. After reading the text, the program deletes the same sentences and below displays a dialogue menu that describes the actions from a particular key pressed in the range from "1" to "4". To exit the program, use the "0" key. To display the current state of the text, you should enter the number "5". In all other cases, the program will display a hint about the incorrectly entered value and ask the user to enter the value again.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: написать программу, которая считывает текст и редактирует его по запросу пользователя.

Для достижения данной цели необходимо решить следующие задачи:

1. Изучить работу с текстом на языке Си.
2. Изучить работу с динамической памятью и научиться использовать функции malloc, calloc, realloc и free.
3. Изучить стандартную библиотеку языка Си.
4. Изучить работу с таким типом данных, как wchar_t, изучить библиотеки <wchar.h> и <wctype.h> и уметь применять их функции.
5. Изучить работу с Makefile.
6. Изучить работу структур и создать структуры для хранения данных текста и предложений.
7. Разработать функции, которые считывают, редактирую и выводят текст, который ввел пользователь.
8. Сгруппировать функции программы по файлам, создать для них заголовочные файлы.
9. Создать Makefile.
10. Протестировать программу.
11. Разбор исключительных ситуаций и исправление ошибок, которые могли появиться в ходе тестирования.

1. ЗАДАНИЕ КУРСОВОЙ РАБОТЫ

Вариант: 6

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 которое встречается не более одного раза в тексте.
2. Каждую подстроку в тексте имеющую вид “<день> <месяц> <год> г.” заменить на подстроку вида “ДД/ММ/ГГГГ”. Например, подстрока “20 апреля 1889 г.” должна быть заменена на “20/04/1889”.
3. Отсортировать предложения по произведению длин слов в предложении.
4. Удалить все предложения, которые содержат символ ‘#’ или ‘№’, но не содержат ни одной 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 main.c

Файл main.c необходим, чтобы были вызваны определённые функции, которые необходимы для реализации выбранного пользователем пункта, описанного в диалоговом окне.

2.2. Файлы dialog.c, dialog.h

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

В файле dialog.c реализована функция, которая даёт пользователю подсказку о том, какое значение нужно ввести, чтобы сделать то или иное действие.

2.3 Файлы txtio.c, txtio.h

В файле txtio.h объявлены функции файла txtio.c. Также в нём объявлены используемые стандартные библиотеки и структуры Sentence и Text.

Структура Text имеет 3 поля: массив, в котором будут хранится элементы типа Sentence, то есть предложения, целочисленная переменная alloc_text — для хранения размера массива, целочисленная переменная sent_count — для хранения количества предложений.

Структура Sentence имеет 4 поля: массив элементов типа wchar_t для хранения предложения, целочисленная переменная symb_count — для хранения количества символов, целочисленная переменная alloc_sent — для хранения размера массива, целочисленная переменная mult_wrd — для хранения произведения слов в предложении.

Также был созданы типы sentence и text от struct Sentence и struct Text соответственно.

В файле txtio.c реализованы 2 функции, которые считывают (input_txt) и выводят (output_txt) текст.

2.4 Файлы txted.c, txted.h

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

В файле txted.c реализованы функции для обработки текста: вывод уникальных слов, изменение даты, сортировка по произведению слов, удаление предложений с определёнными символами и удаление пустых предложений.

2.5 Файлы del_sent.c, del_sent.h

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

В файле del_sent.c реализована функция для удаления всех одинаковых предложений, которые встречаются в тексте, а также реализована вспомогательная функция, которая помогает найти такие предложения.

2.6 Файлы cmp.c, cmp.h

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

В файле cmp.c реализованы 2 функции-компаратора, первая из которых является наиболее общей и сравнивает два объекта и в зависимости от полученного результата возвращает три значения, а вторая сравнивает произведения слов в предложениях.

2.7 Файл Makefile

В Makefile реализована компиляция и линковка всех файлов.

3. ФУНКЦИИ ПРОГРАММЫ

3.1 main()

Задача: главная функция.

- Объявляется функция setlocale для работы с типом данных wchar_t.
- Объявляется переменная text типа text, в которой будет хранится считанный текст.
- Выполняется функция считывания текста input_txt(), которая записывает в text вводимый пользователем текст.
- Выполняется функция удаления одинаковых предложений.
- Выполняется функция output(), выводящая текст.
- Инициализация переменной n, которая будет хранить введённое пользователем значение.
- Цикл while, который вызывает функцию dialog(), которая предоставляет пользователю выбор дальнейшего действия.
- Конструкция switch-case вызывает соответствующие функции в зависимости от того, что ввёл пользователь. (первую (1), вторую (2), третью (3) или четвёртую (4) функции, а также вывод текста при «5», выход из программы при «0» или строку *"Incorrect value entered! Please use the correct entry."*, если пользователем было введено другое значение)
- После ввода пользователем значения «0» программа очищает память при помощи функции free() указателей на предложения, а далее и самого указателя на текст.

3.2 input_txt()

Задача: считать текст.

- Вывод подсказки, инициализация необходимых переменных и выделение памяти под текст при помощи функции malloc(). Переменная checker проверяет, являлся ли считанный символ разделителем, и если да, то

checker равняется 1. Переменная dot, которая будет равна 1 в том случае, если считанный символ окажется точкой.

- Цикл while, считывающий посимвольно текст до тех пор, пока не будет введён пользователем «enter».
    1. Проверка условия, начинается ли предложение с разделителя, если да, то символ пропускается.
    2. Проверка условия на второй подряд разделитель, который не должен быть считан. Иначе будет проверяться не является ли символ концом предложения, а то есть точкой. Если прошлый символ был разделитель, а текущий символ является точкой, то предыдущий символ становится точкой, текущий символ становится символом конца строки, количество символов уменьшается на единицу и записывается в symb_count текущего предложения. Идёт запись символов в следующее предложение.
    3. Выделение памяти для нового предложения.
    4. Символ записывается в предложение.
    5. Если под предложения заканчивается память, то с помощью функции realloc выделяется память ещё под 20 новых предложений.
    6. Если под предложения заканчивается память, то с помощью функции realloc выделяется память ещё под 30 новых символов.
    7. Если считанный символ является точкой, то в конец строки записывается нулевой символ и начинается запись нового предложения.
    8. Если считанный символ является пробелом или запятой, то проверяющая переменная равна 1
- Если была встречена точка, то считывание текста заканчивается. Иначе если предыдущий символ был пробелом, а текущий стал запятой, то на предыдущий символ ставится точка и предложение заканчивается. Иначе на текущий символ ставится точка и следующий символ становится символом конца строки. Количество предложений увеличивается на

единицу и последний символ предложения становится либо символом конца строки, либо точкой.

3.3 del_sent(), in_out()

Задача: удалить одинаковые предложения.

- Инициализация необходимых переменных.
- Выделение памяти под массив arr_num, содержащего номера необходимых к удалению предложений.
- Два цикла for (первый для первого предложения, а второй для второго), в которых сравниваются предложения для быстроты алгоритма сначала по длине с помощью функции wcslen, а далее и по символам с помощью функции wcscasecmp.
    1. Если номера предложений превышают 20, то добавляется место ещё для 20 новых.
    2. Если предложения с индексом ind_sent1 нет в массиве, то оно добавляется.
    3. Если предложения с индексом ind_sent2 нет в массиве, то оно добавляется.
    4. В пунктах 2 и 3 используется вспомогательная функция in_out, которая получает на вход
- Массив номеров сортируется при помощи qsort для быстрого удаления предложений.
- Цикл for:
    1. Переменная shift_num, которая подсчитывает сдвиг предложений и в зависимости от этого удаляет нужные.
    2. Вложенный цикл for удаляет предложение (зануляет все символы в предложении).
    3. Вложенный цикл for меняет все данные следующего предложения на данные предыдущего и смещает предложения. Условие проверяет,

хватает ли памяти для предыдущего предложения, в которое записывается следующее.

4. Уменьшается количество предложений.
5. Переменная-счётчик смещения увеличивается на 1.

- Освобождается память для массива номеров предложений.

3.4 output_txt()

Задача: вывести текст.

- Вывод текста при помощи цикла for.

3.5 dialog()

Задача: считать и вернуть корректное значение

- Инициализация переменной index, которая отвечает за индекс введённого символа.
- Инициализация вводимой строки и выделение ей памяти.
- Инициализация указателя на конец вводимой строки.
- Вывод в консоль диалогового окна.
- Пока не ввёлся символ перевода строки, считываем посимвольно, и если нужно, то выделяем дополнительно памяти для вводимой строки.
- Условия для определения неправильного ввода строки.
- Очищение памяти для строки.
- Функция возвращает итоговое значение

3.6 func_1()

Задача: вывести уникальные слова из введённого текста

- Инициализация всех необходимых переменных.
- Цикл for:
  1. Разделение предложения на слова, исключая все знаки препинания и пробелы.
  2. Выделение достаточного объёма памяти

3. Запись в массив всех слов.

- Цикл for, который пробегается по массиву всех слов и удаляет те слова, которые повторяются.
- Цикл for, который выводит все слова, которые не повторяются.
- Если предыдущий цикл не был выполнен ни разу, то есть не были найдены уникальные слова, то выводится соответствующая строка.
- Вывод завершения функции.
- Освобождение памяти для ненулевых строк.
- Освобождение памяти для переменной, хранящей текущее предложение.
- Освобождение памяти для массива строк.

3.7 func_2()

Задача: найти дату в одном формате и заменить на другой формат

- Инициализация всех необходимых переменных.
- Инициализация массивов:
    1. months — массив месяцев в строковом представлении.
    2. max_days — максимальное количество в каждом месяце по индексам (0 — январь, 1 — февраль и т. д.)
    3. month_symbs — максимальное количество символов каждого месяца строкового представления.
- Цикл for, который пробегается по всем предложениям и ищет дату:
    1. Цикл for, который находит в предложении дату, а если не находит, то завершается и переходит.
    2. Условие перехода к другому предложению, если в этом не было обнаружено совпадений.
    3. Условие нахождения нужного формата для обработки.
    4. Условие нахождения даты формата «1 апреля», «5 марта» и т. д.
    5. Условие, которое определяет, является ли введённый год високосным.
    6. Условие определения подходящих под реальные числовые данные.
    7. Условие преобразования даты к нужному формату.

- Вывод завершения функции.

3.8 func_3()

Задача: отсортировать предложения в тексте по произведению длин слов.

- Инициализация необходимых переменных.
- Цикл for, который пробегается по каждому предложению и в котором проверяется, состоит ли предложение из одного слова, и если да, то присваивается произведению длин слов в этом предложении 0, в другом же случае берёт по отдельности каждое слово и считает количество символов, которое умножает на переменную n.
- Быстрая сортировка qsort сортирует предложения по произведению длин слов в предложении.
- Вывод завершения функции.
- Освобождение памяти для предложения.

3.9 func_4()

Задача: удалить предложения, в которых встречается символ № или #, но не встречается ни одной цифры.

- Инициализация всех необходимых переменных.
- Цикл for, который пробегается по каждому предложению и в котором проверяется условие, входит ли в предложение символ № или # и нет ли в предложении цифр. Если условие подходит, то предложение удаляется.
- Выполняется функция del_void, которая удаляет пустые предложения.
- Освобождается переменная sent.
- Вывод завершения функции.

3.10 del_void()

Задача: удалить пустые предложения.

- Инициализация всех необходимых переменных.

- Цикл while, который ищет пустое предложение, сдвигает все остальные предложения, меняет данные предложений и выделяет необходимый объём данных для сдвигаемых предложений.

3.11 cmp() и cmp_txt()

Задача: сравнить два объекта для общего случая или произведение количества символов слов в предложениях

Функции компараторы, которые принимают и сравнивают два объекта.

4. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

4.1. Считывание и вывод текста.

Входные данные: *Здравствуйте, мистер. Вы ввели лишние точки в конце предложения..... И ушли очень далеко.*

Выходные данные: *Здравствуйте,мистер.Вы ввели лишние точки в конце предложения.И ушли очень далеко.*

Программа успешно убрала лишние знаки препинания и пробелы.

4.2. Удаление одинаковых предложений.

Входные данные: Ау-ау, ищи меня. Ты не найдёшь меня. Ау-ау, ищи меня. Я исчез.

Выходные данные: Ты не найдёшь меня.Я исчез.

Программа удалила из текста первое и третье одинаковые предложения.

4.3 Вывод уникальных слов.

Входные данные: Уникальные слова. Слова уникальны. Только не эти, друг мой. Только, друг мой, не эти.

Выходные данные:

Уникальные

слова

Слова

уникальны

Программа вывела в каждой строке слово, которое не встречается в тексте более 1 раза.

4.4 Замена разных форматов даты.

Входные данные: Мой день рождения 24 января 2000 г. Сегодня 21 декабря 2019 г. Новый год будет 01 января 2020 г. Нового года не будет 1 февраля 2020 г.

Выходные данные: Мой день рождения 24/01/2000.Сегодня 21/12/2019.Новый год будет 01/01/2020.Нового года не будет 1 февраля 2020 г.

Особенность программы в том, что дата обязательно является концом предложения, так как в конце формата встречается точка (г.). Поэтому, чтобы можно было отличить предложения друг от друга, была вставлена точка после формата даты «ДД/ММ/ГГГГ». Для корректной работы необходимо вводить день как 01, 02 и т. д.

4.5 Сортировка предложений по произведению длин слов в них.

Входные данные: 0000. 0016 0016. 04 04. 009 009.

Выходные данные: 0000.04 04.009 009.0016 0016.

Числа означают числовое значение произведения длин слов в этих предложениях

4.6 Удаление предложений, в которых содержится символ № или #, но не содержится ни одной цифры.

Входные данные: Мой № телефона +794632154**. А вот его № я не помню. Ещё есть обозначение с #. Но номер #7974******* сейчас не доступен.

Выходные данные: *Мой № телефона +794632154**.Но номер #7974******* сейчас не доступен.*

ЗАКЛЮЧЕНИЕ

Были изучены необходимые библиотеки языка Си. Реализованы функции, которые позволяют считывать и редактировать текст исходя из пожеланий пользователя. Функции находятся в нужных файлах, также для каждого файла .c создан заголовочный файл. Реализован Makefile для компиляции и линковки проекта. Обработаны исключительные ситуации, которые могут возникнуть при неправильном вводе текста. Предусмотрен вывод текста, если пользователь введёт «5», также предусмотрен выход пользователя из программы при вводе значения «0».

ПРИЛОЖЕНИЕ А

РАЗРАБОТАННЫЙ КОД

1. Файл main.c:

```
#include "txtio.h"
#include "txted.h"
#include "del_sent.h"
#include "dialog.h"

int main(){
    setlocale(LC_ALL,"");
    text text;
    input_txt(&text);
    del_sent(&text);
    output_txt(&text);
    int n;
    while ((n = dialog()) != 0){
        switch (n){
            case 1:
                func_1(&text);
                break;
            case 2:
                func_2(&text);
                break;
            case 3:
                func_3(&text);
                break;
            case 4:
                func_4(&text);
                break;
            case 5:
                output_txt(&text);
                break;
            default:
                fputws(L"Incorrect value entered! Please use the correct entry.\n", stdout);
                break;
        }
    }
    for (int i = 0; i < text.sent_count; i++){
        free(text.text[i].words_arr);
    }
    free(text.text);
    return 0;
}
```

2. Файл dialog.h:

```
#ifndef DIALOG_H
#define DIALOG_H

#include "txtio.h"

int dialog();

#endif
```

3. Файл dialog.c:

```
#include "dialog.h"
#define FAULT 666

int dialog(){
   int index = 0, enter_size = 20, out = 0;
   wchar_t* string = (wchar_t*)calloc(enter_size, sizeof(wchar_t));
   wchar_t* endptr;
   fputws(L"\nEnter one of the following values for the action:\n", stdout);
   fputws(L"0 (zero) - Complete the program\n", stdout);
   fputws(L"1 - Print each word that occurs no more than once in the text.\n", stdout);
    fputws(L"2 - Each substring in the text having the form “<день> <месяц> <год> г.” is replaced by a substring of the form “ДД / ММ / ГГГГ”.\n", stdout);
     fputws(L"3 - Sort sentences by product of lengths of words in a sentence.\n", stdout);
    fputws(L"4 - Delete all sentences that contain the symbol ‘#’ or ‘№’ but do not contain a single digit.\n", stdout);
   fputws(L"5 - Display the text\n", stdout);
   fputws(L"Enter the value: ", stdout);
   while ((string[index] = getwchar()) != '\n'){
      if (index == enter_size - 1){
         enter_size += 20;
         string = (wchar_t*)realloc(string, enter_size*sizeof(wchar_t));
      }
      index++;
   }
   fputws(L"\n", stdout);
   if (string[0] == '\n'){
      free(string);
      return FAULT;
   }
   string[index] = '\0';
   for (int i = 0; string[i] != '\0'; i++)
```

```
        if (iswdigit(string[i])) continue;
        else{
          free(string);
          return FAULT;
        }
    out = (int)wcstol(string, &endptr, 10);
    free(string);
    return out;
}
```

4. Файл txtio.h

```
#ifndef TXTED_H
#define TXTED_H

#include "cmp.h"
#include "txtio.h"

void func_1(text* text);
void func_2(text* text);
void func_3(text* text);
void func_4(text* text);
void del_void(text* text);

#endif
```

5. Файл txtio.c:

```
#include "txtio.h"

void input_txt(struct Text *text){
    fputws(L"Please enter the text: \n",stdout);
    int checker = 0, dot = 0, ind_sent = 0, ind_symb = 0;
    wchar_t symbol;
    text->alloc_text = 10;
    text->text = (sentence*)malloc(sizeof(sentence)*text->alloc_text);
    text->sent_count = 0;
    while ((symbol = getwchar()) != '\n'){
          if ((symbol == L' ' || symbol == L'.' || symbol == ',' ) && ind_symb == 0)
continue;
        if (checker == 1 && (symbol == L',' || symbol == ' ')) continue;
        else if (checker == 1 && symbol == '.'){
            text->text[ind_sent].words_arr[ind_symb-1] = symbol;
            text->text[ind_sent].words_arr[ind_symb] = '\0';
            text->text[ind_sent++].symb_count = ind_symb-1;
            ind_symb = 0;
```

```
            dot = 1;
            text->sent_count++;
            continue;
        }
        checker = 0;
        if (ind_symb == 0){
            text->text[ind_sent].alloc_sent = 50;
                text->text[ind_sent].words_arr = (wchar_t*)malloc(sizeof(wchar_t)*(text-
>text[ind_sent].alloc_sent));
        }
        text->text[ind_sent].words_arr[ind_symb++] = symbol;
        dot = 0;
        if (ind_sent == text->alloc_text - 1){
            text->alloc_text += 20;
                        text->text = (sentence*)realloc(text->text, sizeof(sentence)*(text-
>alloc_text));
        }
        if(ind_symb == text->text[ind_sent].alloc_sent - 3){
            text->text[ind_sent].alloc_sent += 30;
                                text->text[ind_sent].words_arr = (wchar_t*)realloc(text-
>text[ind_sent].words_arr, sizeof(wchar_t)*(text->text[ind_sent].alloc_sent));
        }
        if (symbol == '.'){
            text->text[ind_sent].words_arr[ind_symb] = '\0';
            text->text[ind_sent++].symb_count = ind_symb;
            text->sent_count += 1;
            ind_symb = 0;
            dot = 1;
        }
        else if (symbol == ' ' || symbol == ','){
          checker = 1;
        }
    }
    if (dot == 0){
                        if (text->text[ind_sent].words_arr[ind_symb-1] == ' ' || text-
>text[ind_sent].words_arr[ind_symb] == ','){
            text->text[ind_sent].words_arr[ind_symb-1] = '.';
            text->text[ind_sent].words_arr[ind_symb] = '\0';
        }
        else{
            text->text[ind_sent].words_arr[ind_symb] = '.';
            text->text[ind_sent].words_arr[ind_symb+1] = '\0';
        }
        text->sent_count++;
```

```
        text->text[ind_sent].symb_count = ind_symb;
    }

}

void output_txt(text *text){
    fputws(L"\nYour text is:\n", stdout);
    for(int i = 0 ; i < text->sent_count; i++){
        fputws(text->text[i].words_arr, stdout);
    }
    fputws(L"\n", stdout);
}
```

6.Файл txted.h:

```
#ifndef TXTED_H
#define TXTED_H

#include "cmp.h"
#include "txtio.h"

void func_1(text* text);
void func_2(text* text);
void func_3(text* text);
void func_4(text* text);
void del_void(text* text);

#endif
```

7. Файл txted.c:

```
#include "txted.h"
#define MONTHS 12

void func_1(text* text){
    wchar_t** arr, *sent = NULL, *state, *str;
    int n = 0, arr_size = 20, no_words = 0;

    arr = (wchar_t**)calloc(sizeof(wchar_t*), arr_size);

    for(int i = 0; i < text->sent_count; i++){
        sent = (wchar_t*)calloc(sizeof(wchar_t), text->text[i].alloc_sent);
        wcscpy(sent, text->text[i].words_arr);

         for (str = wcstok(sent, L" ,.", &state); str != NULL; str = wcstok(NULL, L" ,.",
&state)){
```

```
            if (n == arr_size - 2){
                arr_size += 20;
                arr = (wchar_t**)realloc(arr, sizeof(wchar_t*)*arr_size);
            }
            arr[n] = (wchar_t*)calloc(sizeof(wchar_t), (wcslen(str)) + 1);
            wcscpy(arr[n], str);
            n++;
        }
    }

    for (int j = 0; j < n - 1; j++){
        str = arr[j];
        if (wcscmp(str, L"") != 0)
            for (int k = j + 1; k < n; k++){
                if (wcscmp(str, arr[k]) == 0){
                    arr[k] = L"";
                    arr[j] = L"";
                }
            }
    }

    for (int i = 0 ; i < n; i++)
    {
        if (wcscmp(arr[i], L"") != 0){
            fputws(arr[i], stdout);
            fputws(L"\n", stdout);
            no_words++;
        }
    }
    if (no_words == 0)
        fputws(L"No unique words in the text\n", stdout);
    fputws(L"\nThe func_1 is done.\n", stdout);

    for (int i = 0 ; i < n; i++){
        if (wcscmp(arr[i], L"") != 0)
            free(arr[i]);
    }
    free(sent);
    free(arr);
}

void func_2(text* text){
     wchar_t *months[MONTHS] = {L"января", L"февраля", L"марта", L"апреля",
L"мая", L"июня",
```

```
                        L"июля", L"августа",L"сентября", L"октября", L"ноября",
L"декабря"};
   int max_days[MONTHS] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
   int month_symbs[MONTHS] = {6, 7, 5, 6, 3, 4, 4, 7, 8, 7, 6, 7};
   int month_num = 0;
   wchar_t *checker = NULL;
   int date, year, leap;

   for (int sent = 0; sent < text->sent_count; sent++){
      leap = 0;
      for (month_num = 0; month_num < MONTHS; month_num++)
            if ((checker = wcsstr(text->text[sent].words_arr, months[month_num])) !=
NULL)
            break;
      if (checker == NULL) continue;

               if (!((iswdigit(*(checker - 3)) && iswdigit(*(checker - 2)) &&
iswspace(*(checker + month_symbs[month_num]))
                && iswdigit(*(checker + month_symbs[month_num] + 1)) &&
iswdigit(*(checker + month_symbs[month_num] + 2))
                && iswdigit(*(checker + month_symbs[month_num] + 3)) &&
iswdigit(*(checker + month_symbs[month_num] + 4))
        && iswspace(*(checker + month_symbs[month_num] + 5)) && (*(checker +
month_symbs[month_num] + 6) == L'r')
      && (*(checker + month_symbs[month_num] + 7) == L'.')))) continue;
      if (!(iswdigit(*(checker - 3)))) date = (int)wcstol(checker - 2, NULL, 10);
      else date = (int)wcstol(checker - 3, NULL, 10);
      year = (int)wcstol(checker + month_symbs[month_num] + 1, NULL, 10);
      if (year % 4 == 0) leap = 1;
        if (!(((date >= 1) && (date <= max_days[month_num])) || ((leap == 1) &&
(month_num == 1)
      && (date == 29))) && (year >= 0)) continue;

      if (month_num + 1 < 10){
           if (year < 10) swprintf(checker - 1, 10, L"/0%d/000%d.", month_num + 1,
year);
          else if (year < 100) swprintf(checker - 1, 10, L"/0%d/00%d.", month_num +
1, year);
          else if (year < 1000) swprintf(checker - 1, 10, L"/0%d/0%d.", month_num +
1, year);
          else if (year < 10000) swprintf(checker - 1, 10, L"/0%d/%d.", month_num +
1, year);
      }
      else{
```

```
            if (year < 10) swprintf(checker - 1, 10, L"/%d/000%d.", month_num + 1,
year);
          else if (year < 100) swprintf(checker - 1, 10, L"/%d/00%d.", month_num + 1,
year);
          else if (year < 1000) swprintf(checker - 1, 10, L"/%d/0%d.", month_num + 1,
year);
          else if (year < 10000) swprintf(checker - 1, 10, L"/%d/%d.", month_num + 1,
year);
       }
   }
   fputws(L"The func_2 is done.\n", stdout);
}

void func_3(text* text){
   int words = 0, n = 1;
   wchar_t *sent = NULL, *str, *state;
   for (int i = 0; i < text->sent_count; i++){
      sent = (wchar_t*)calloc(sizeof(wchar_t), text->text[i].alloc_sent);
      wcscpy(sent, text->text[i].words_arr);

       for (str = wcstok(sent, L" ,.", &state); str != NULL; str = wcstok(NULL, L" ,.",
&state)) {
          n *= (int) wcslen(str);
          words++;
       }
      if (words == 1) text->text[i].mult_wrd = 0;
      else text->text[i].mult_wrd = n;
      n = 1;
      words = 0;
   }
   qsort(text->text, text->sent_count, sizeof(sentence), cmp_txt);
   fputws(L"The func_3 is done.\n", stdout);
   free(sent);
}

void func_4(text* text){
   int lat, num;
   wchar_t *sent = NULL;
   for (int i = 0; i < text->sent_count; i++){
      sent = (wchar_t*)calloc(sizeof(wchar_t), text->text[i].alloc_sent);
      wcscpy(sent, text->text[i].words_arr);
      if (wcschr(sent, L'#')) lat = 1;
      else lat = 0;
      if (wcschr(sent, L'№')) num = 1;
```

```
        else num = 0;
            if (!((wcschr(sent, L'0')) || (wcschr(sent, L'1')) || (wcschr(sent, L'2')) ||
(wcschr(sent, L'3')) ||
        (wcschr(sent, L'4')) || (wcschr(sent, L'5')) || (wcschr(sent, L'6')) || (wcschr(sent,
L'7')) ||
        (wcschr(sent, L'8')) || (wcschr(sent, L'9'))) && (lat != num)){
            for (int j = 0; j <= text->text[i].symb_count; j++){
                text->text[i].words_arr[j] = L'\0';
            }
        }
    }
    del_void(text);
    free(sent);
    fputws(L"The func_4 is done.\n", stdout);
}

void del_void(text* text){
    int count;
    int i = 0;
    while(i < text->sent_count){
        if (wcscmp(text->text[i].words_arr, L"") == 0){
            count = text->sent_count - 1;
            for(int j = i; j < count; j++){
                if (text->text[j].alloc_sent < text->text[j+1].alloc_sent){
                        text->text[j].words_arr = (wchar_t*)realloc(text->text[j].words_arr,
sizeof(wchar_t)*text->text[j+1].alloc_sent);
                    text->text[j].alloc_sent = text->text[j+1].alloc_sent;
                }
                wcscpy(text->text[j].words_arr,text->text[j+1].words_arr);
                text->text[j].symb_count = text->text[j+1].symb_count;
            }
            text->sent_count -= 1;
            continue;
        }
        i++;
    }
}
```

8. Файл del_sent.h:

```
#ifndef DEL_SENT_H
#define DEL_SENT_H

#include "txtio.h"
```

```
void del_sent (text *text);
int in_out (int*arr_num, int num, int ind_sent);

#endif
```

9. Файл del_sent.c:

```
#include "del_sent.h"
#include "cmp.h"

void del_sent (text* text)
{
    int len_arr = 20, num = 0, shift = 0, ind_sent1 = 0, ind_sent2 = 0, sent_arr_num,
shift_num, *arr_num;
   arr_num = (int*)malloc(sizeof(int)*len_arr);
   for (ind_sent1 = 0; ind_sent1 < text->sent_count; ind_sent1++)
      for (ind_sent2 = ind_sent1 + 1; ind_sent2 < text->sent_count; ind_sent2++)
                        if (wcslen(text->text[ind_sent1].words_arr) == wcslen(text-
>text[ind_sent2].words_arr))
                              if (wcscasecmp(text->text[ind_sent1].words_arr, text-
>text[ind_sent2].words_arr) == 0){
                if (num >= len_arr - 2){
                   len_arr += 20;
                   arr_num = realloc(arr_num, sizeof(int)*len_arr);
                }
                if (in_out(arr_num, num, ind_sent1)){
                   arr_num[num] = ind_sent1;
                   num++;
                }
                if (in_out(arr_num, num, ind_sent2)){
                   arr_num[num] = ind_sent2;
                   num++;
                }
             }
   qsort(arr_num, num, sizeof(int), cmp);
   for (ind_sent1 = 0; ind_sent1 < num; ind_sent1++){
      shift_num = arr_num[ind_sent1] - shift;
      for (int j = 0; text->text[shift_num].words_arr[j]; j++){
         text->text[shift_num].words_arr[j] = '\0';
      }
      sent_arr_num = text->sent_count - 1;
      for (ind_sent2 = shift_num; ind_sent2 < sent_arr_num; ind_sent2++){
         if(text->text[ind_sent2].alloc_sent < text->text[ind_sent2+1].alloc_sent){
              text->text[ind_sent2].words_arr = realloc(text->text[ind_sent2].words_arr,
sizeof(wchar_t)*text->text[ind_sent2+1].alloc_sent);
```

```
            }
            text->text[ind_sent2].symb_count = text->text[ind_sent2+1].symb_count;
            text->text[ind_sent2].alloc_sent = text->text[ind_sent2+1].alloc_sent;
            wcscpy(text->text[ind_sent2].words_arr, text->text[ind_sent2+1].words_arr);
        }
        text->sent_count -= 1;
        shift += 1;
    }
    free(arr_num);
}

int in_out (int* arr_num, int num, int ind){
    for(int i = 0 ; i < num; i++){
        if (arr_num[i] == ind){
            return 0;
        }
    }
    return 1;
}
```

10. Файл cmp.h

```
#ifndef CMP_H
#define CMP_H

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

#endif
```

11.Файл cmp.c

```
#include "txtio.h"

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

int cmp_txt(const void* a, const void* b){
    if (((sentence*)a)->mult_wrd > ((sentence*)b)->mult_wrd)
        return 1;
    if (((sentence*)a)->mult_wrd < ((sentence*)b)->mult_wrd)
```

```
        return -1;
    return 0;
}
```

12. Файл Makefile

```
main: main.o txtio.o del_sent.o txted.o dialog.o cmp.o
	gcc main.o txtio.o del_sent.o txted.o dialog.o cmp.o -o main

main.o: main.c
	gcc -c main.c

txtio.o: txtio.c
	gcc -c txtio.c

del_sent.o: del_sent.c
	gcc -c del_sent.c

txted.o: txted.c
	gcc -c txted.c

dialog.o: dialog.c
	gcc -c dialog.c

cmp.o: cmp.c
	gcc -c cmp.c

clean:
	rm *.o
```