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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Работа со строками

Студентка гр. 9382 _____ Круглова В. Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студентка Круглова В. Д.

Группа 9382

Тема работы: Работа со строками.

Исходные данные: программа должна быть модульной, обработка данных должна производится при возможности с использованием функций стандартных библиотек языка. Сохранение, обработка и вывод данных должны осуществляться с помощью работы со структурами и динамической памятью.

Содержание пояснительной записки:

«Содержание», «Введение», «Задание работы», «Ход выполнения работы» «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 15.10.2019

Дата сдачи реферата:

Дата защиты реферата:

Студентка гр. 9382 _____ Круглова В. Д.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

Данная курсовая работа по обработке строк, которая представляет собой их ввод, вывод и форматирование, выполнена с использованием структур, указателей, произвольных динамических массивов и функций стандартных библиотек. Для упрощения обращения к программе был написан Makefile, при создании заголовочных файлов была предусмотрена защита от повторного обращения к файлам. Также было красиво оформлено меню, из которого предусмотрен выход и даже прощание с пользователем.

Пример работы программы приведён в приложении А.

SUMMARY

This course work on processing strings, which means their input, output and formatting, is performed using structures, pointers, arbitrary dynamic arrays and functions of standard libraries. To simplify access to the program, a Makefile was written, while creating header files, protection was provided against repeated access to files. The menu was also beautifully designed, from which an exit and even a farewell to the user was provided.

An example of the program is given in annex A.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы: написать код программы, способной в соответствии с желанием пользователя выполнять некоторое количество функций, представленных в меню, которое предусматривает выход из программы. Должны быть использованы структуры данных, указатели, одномерные и двумерные массивы; любые написанные функции должны быть реализованы при помощи стандартных библиотек, где это можно сделать; должна быть возможность использования русского языка при работе с программой; должен быть написан Makefile.

Программа была разработана на базе ОС Linux в Visual Studio Code, запуск производился через терминал. Отладка производилась вручную при помощи дополнительных источников информации.

Итак, написанная программа запрашивает и считывает произвольной длины текст, который пользователь вводит с консоли, затем выводит меню с названиями предложенных функций и выполняет выбранную пользователем, после чего снова выводит меню, пока пользователь не решит выйти из нее. Итогом работы программы является обработанный текст, после успешного выполнения она прощается с пользователем.

1. ЗАДАНИЕ

ВАРИАНТ 14

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):

Посчитать и вывести в минутах количество секунд встречающихся в тексте. Количество секунд задается в виде подстроки “ <число> sec “.
Отсортировать предложения по увеличению сумме кодов символов первого слова в предложении..

Заменить все символы ‘%’, ‘#’, ‘@’ на “<percent>”, “<решетка>”, “(at)” соответственно.

Удалить все предложения которые заканчиваются на слово с тремя согласными подряд.

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. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Создание структур и заголовочных файлов.

Были созданы две структуры Sentence и Text в файле IncludeAll.h с использованием предохранения от переобьявления: #ifndef #endif. Также выполнен typedef для более удобного обращения к структурам.

Содержание структур: в Sentence содержится переменная CountSymbols типа int для подсчета колличества символов в каждом предложении. аналогично MaxCountSymbols для выделения памяти на эти символы; NewCount – аналогично CountSymbols, но для массива измененных предложений и MaxNewCount для выделения памяти под него. Также одномерные массивы sentence и ChangeSentence типа wchar_t для хранения исходных и измененных предложений соответственно. Что касается структуры Text, в ней создается динамический массив text типа struct Sentence. Там же определяюся счетчики колличества предложений исходного и измененного текстов (CountSentence и MaxCountSentence) и массив букв – word, в который будет замисан введенный текст.

Содержание заголовочных файлов: в файлач InputText.h и Funcs.h содержатся функции для ввода, вывода и удаления повторяющихся предлжений (в первом файле) и функции по обработки текста.

2.2. Чтение и заведение в структуру текста.

С помощью функции fgetws, адаптированной и под кириллицу заполняется массив одномерный word, на который предварительно выделена память и в процессе ввода ее перераспределяли с помощью функций malloc и

realloc в файле InputText.c. Затем посимвольно перенесен текст, находящийся в массиве word в структуру, после чего память выделенная на этот массив очищается.

Функция input принимает указатель на динамический массив txt типа struct Text и в результате заполняет этот массив, ничего при этом не возвращая.

Введенные данные записываются в txt, где каждое предложение отделено от предыдущего символом окончания строки.

2.3. Первоначальная обработка текста.

После записи текста в структуру в функции CleanerSame того же файла удаляются все повторяющиеся предложения с помощью функции free.

Она сравнивает все предложения между собой и, если находит одинаковые по длине, проверяет в них каждый символ на идентичный с учетом номера символа, предварительно переводя их в верхний регистр. Чтобы отследить схожесть предложений, используется счетчик f. При удалении предложения все следующие двигаются на одно влево.

На вход функции подается указатель на текст, сама функция ничего не выводит, но в итоге преобразует текст.

2.4. Функции из меню.

Здесь будет поясняться работа функций файла Funcs.c

2.4.1 Замена символов «@, %, #».

Функция change принимает на вход указатель на текст, после чего проходится по предложениям и ищет вышеперечисленные символы с помощью функции wcsstr, которая возвращает указатель на искомый символ, и, если хоть один из символов присутствует в предложении, запускает алгоритм, сначала выделяющий память на новое предложение, потом в процессе ее перераспределяя, далее посимвольно проверяет предложения на наличие искомых. В случае нахождения какого-либо из них перераспределяет память и на место символа записывает первый символ строки, на которую надо

заменить, потом увеличивает номер символа в измененном предложении и записывает следующий символ, пока не сделает полную замену, потом увеличивает то количество еще раз и обрабатывает следующий символ. Если он не входит в искомый перечень, записывается в измененное предложение и снова увеличивается счетчик их количества. Когда все символы исходного предложения были обработаны, в конце измененного предложения ставится символ окончания строки, затем исходное предложение очищается, а на место него записывается измененное.

На вход передается указатель на текст, в результате работы получаем измененный текст.

2.4.2 Сортировка по возрастанию кода первого слова предложений.

Функция sort сортирует предложения, находящиеся в text, размера CountSentence используя компоратор, который сначала ищет первое слово в предложении и считает сумму кодов его символов, после чего сравнивает получившиеся количества и передает эти данные в функцию сортировки, определяя порядок, в котором эти предложения будут расположены.

Функция sort принимает указатель на текст и в процессе работы изменяет его, ничего при этом не возвращая.

2.4.3 Удаление предложений, оканчивающихся на 3 согласные.

Функция del проходит по всем предложениям и проверяет символы начиная с конца на то согласная это или нет, в случае, если это вдруг не согласная, захождение в условие удаления, когда количество этих согласных, перечисленных в массиве в начале функции и сравниваемых далее в цикле for, равно 3 становится невозможно. Если же программа все-таки заходит в это условие, с помощью функции free очищает это предложение и двигает все следующие влево на 1 позицию.

Функция del принимает указатель на текст и в процессе работы изменяет его, ничего при этом не возвращая.

2.4.4 Подсчет минут.

Функция CounterMin принимает указатель на текст, проходит по символам предложений переданного текста и ищет подряд идущие буквы „s“, „e“, „c“ и пробел перед первой. Если находит, то читается и переводится в int с помощью функции atoi и умножается на 10 в степени равной порядковому номеру с конца числа (индексация с 0). Затем к счетчику прибавляется это число.

Функция выводит это количество, переведенное в минуты (с округлением до целых).

Особенностью работы является требование ко вводу в формате <число> sec, иначе программа не засчитает это как источник информации о количестве секунд.

2.5 Вывод текста.

Функция output, получающая на вход указатель на текст, выводящая текст располагается в файле inputText.c, она осуществляет вывод текста по предложениям.

В дальнейшем она будет использована для вывода в после осуществления некоторых функций меню.

2.6 Основная часть программы.

В данном разделе освещаются главные аспекты файла main.c, к которому подключены все заголовочные файлы.

2.6.1 Общение с ползователем.

При запуске программа выводит сообщение о готовности считывать текст. После ввода и отправления текста пользователем выводится меню, возможность выхода из которого предусмотрена нажатием любой клавиши кроме уже используемых. Затем программа прощается с пользователем.

2.6.2 Освобождение памяти.

Когда пользователь выбирает в меню выход из программы, оператор switch действует по дефолту и проходит по всем предложениям текста, очищая их одно за другим, затем очищая ячейки текста, соответствующие этим предложениям.

После чего очищается и вся переменная, содержащая текст. Также предусматривается невозможность дальнейшего обращения к меню.

2.7 Создание Makefile.

Все созданные файлы связаны зависимостями компиляции в файле Makefile, также выявлены корректные цели для каждого бинарного файла. Все заголовочные файлы были защищены от переопределения. Он дает возможность скомпилировать и запустить программу, введя в терминал команду make.

ЗАКЛЮЧЕНИЕ

В процессе создания программы были применены знания по работе со строками, функциями стандартных библиотек, простейшим алгоритмам взаимодействия со структурами, а также использования указателей и динамических массивов. Усовершенствованы навыки в создании Makefile.

Итогом работы является работающая программа, готовая в соответствии с введенными данными и требованиями пользователя выполнять доступные ей функции. Программа работает корректно при правильном вводе с консоли, тонкости ввода можно найти в описании функций меню или спросить про понимание задания у исполнителя лично.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

ПРИЛОЖЕНИЕ А

ПРИМЕР РАБОТЫ ПРОГРАММЫ

Пример вывода приветствия пользователя и ввод им текста (An example of the output of user greetings and text input):

```
>> Input your text.
>>An example of the output of user greetings and text input
 >> Enter 5 for more information
>> Choose command:
>>2
 >> An example of the output of user greetings and text input.
>>Choose command:
>>0
 >> An example of the output of user greetings and text input.
>> Choose command:
>>9
 >> See you.
Process finished with exit code 0
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД ПРОГРАММЫ

название файла: main.c

```
#include "IncludeAll.h"
#include "inputText.h"
#include "Funcs.h"


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

wprintf(L"////////////////////////////////////////////////////////
//////////////////////////////////////////////////\n"
            L"\t\t\t\t\tInput your text.\n"

L"////////////////////////////////////////////////////////////////
//////////////////////////////////////////\n");
    input(&txt);
    InputInStruct(&txt);
    CleanerSame(&txt);
    int NumberOfFunction = 0;
    int bool;
    while (NumberOfFunction != 8){

wprintf(L"////////////////////////////////////////////////////////
//////////////////////////////////////////////////\n"
                "\tChoose command:\n"
                L"\tPress 0, to output all text.\n"
                L"\tPress 1, to count minutes.\n"
                L"\tPress 2, to sort.\n"
                L"\tPress 3, to change @, %, #.\n"
                L"\tPress 4, to delete sentences.\n"
                L"\tPress else to close.\n"

"////////////////////////////////////////////////////////////////
//////////////////////////////////////////\n");
        bool = wscanf(L"%d", &NumberOfFunction);
```

```
        if (bool != 1){
            NumberOfFunction = 8;
        }
        switch (NumberOfFunction) {
            case 0:
                output(&txt);
                break;
            case 1:
                CounterMin(&txt);
                break;
            case 2:
                sort(&txt);
                output(&txt);
                break;
            case 3:
                change(&txt);
                output(&txt);
                break;
            case 4:
                del(&txt);
                output(&txt);
                break;
            default:
                for(int i = 0; i < txt.CountSentence; i++){
                    free(txt.text[i]->sentence);
                    free(txt.text[i]);
                }
                free(txt.text);
                wprintf(L"See you.\n");
                NumberOfFunction = 8;
                break;
        }
    }
    return 0;
}
```

название файла inputText.c:

```
#include "inputText.h"

#define ARRAY_SIZE 32
```

```
void input(Text* txt) {
    txt->word = malloc(sizeof(wchar_t) * ARRAY_SIZE);
    fgetws(txt->word, ARRAY_SIZE, stdin);
    while (wcschr(txt->word, L'\n') == NULL) {
        txt->word = realloc(txt->word, sizeof(wchar_t) *
(wcslen(txt->word) + ARRAY_SIZE));
        fgetws(txt->word + wcslen(txt->word), ARRAY_SIZE, stdin);
    }
}
void InputInStruct(Text* txt){
    txt->CountSentence = 0;
    int bool = 1;
    txt->MaxCountSentence = ARRAY_SIZE;
    txt->text = malloc(ARRAY_SIZE * sizeof(Sentence*));
    for (int i = 0; txt->word[i] != '\n'; i++){
        if (txt->CountSentence == txt->MaxCountSentence){
            txt->MaxCountSentence += ARRAY_SIZE;
            txt->text = realloc(txt->text, txt->MaxCountSentence *
sizeof(Sentence*));
        }
        if (bool == 1){
            txt->text[txt->CountSentence] =
malloc(sizeof(Sentence));
            txt->text[txt->CountSentence]->CountSymbols = 0;
            txt->text[txt->CountSentence]->MaxCountSymbols =
ARRAY_SIZE;
            txt->text[txt->CountSentence]->sentence = malloc( txt-
>text[txt->CountSentence]->MaxCountSymbols *

sizeof(wchar_t));
            bool = 0;
        }
        if ( txt->text[txt->CountSentence]->MaxCountSymbols ==
txt->text[txt->CountSentence]->CountSymbols){
            txt->text[txt->CountSentence]->MaxCountSymbols +=
ARRAY_SIZE;
            txt->text[txt->CountSentence]->sentence =
realloc( txt->text[txt->CountSentence]->sentence,  txt->text[txt-
>CountSentence]->MaxCountSymbols *
```

```
sizeof(wchar_t));
        }
        if(txt->word[i] != '.'){
            if(((txt->word[i] == ' ') || (txt->word[i] == ',')) &&
(txt->text[txt->CountSentence]->CountSymbols == 0)){
                continue;
            }
            txt->text[txt->CountSentence]->sentence[txt->text[txt-
>CountSentence]->CountSymbols] = txt->word[i];
            txt->text[txt->CountSentence]->CountSymbols++;
        }
        else{
            if( txt->text[txt->CountSentence]->CountSymbols == 0){
                continue;
            }
            txt->text[txt->CountSentence]->sentence =
realloc( txt->text[txt->CountSentence]->sentence,  (txt->text[txt-
>CountSentence]->MaxCountSymbols + 1) *

sizeof(wchar_t));
            txt->text[txt->CountSentence]->sentence[txt->text[txt-
>CountSentence]->CountSymbols] = '.';
            txt->text[txt->CountSentence]->CountSymbols++;
            txt->text[txt->CountSentence]->sentence[txt->text[txt-
>CountSentence]->CountSymbols] = '\0';
            txt->CountSentence++;
            bool = 1;

        }
    }
    free(txt->word);
}

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

```
void CleanerSame(Text* txt){
    for(int i = 0; i < txt->CountSentence; i++){
        for(int j = 0; j < txt->CountSentence - 1; j++){
            if (i == j){
                continue;
            }
            else{
                if(txt->text[i]->CountSymbols == txt->text[j]-
>CountSymbols){
                    char f = 1;
                    for(int k = 0; k < txt->text[i]->CountSymbols;
k++){
                        if (towupper(txt->text[i]->sentence[k]) !=
towupper(txt->text[j]->sentence[k])){
                            f = 0;
                        }
                    }
                    if (f == 1){
                        free(txt->text[j]);
                        txt->CountSentence--;
                        for (int k = j; k < txt->CountSentence; k+
+){
                            txt->text[k] = txt->text[k+1];
                        }
                        j--;
                    }
                }
            }
        }
    }
}
```

название файлы Funcs.c:

```
#include "Funcs.h"
#define ARRAY_SIZE 32

void change(Text* txt){
    for(int i = 0; i< txt->CountSentence; i++) {
```

```
    if (wcsstr(txt->text[i]->sentence, L"@") != NULL ||
wcsstr(txt->text[i]->sentence, L"#") != NULL ||
            wcsstr(txt->text[i]->sentence, L"%") != NULL) {
            txt->text[i]->MaxNewCount = ARRAY_SIZE;
            txt->text[i]->ChangeSentence = malloc(txt->text[i]-
>MaxNewCount * sizeof(wchar_t));
            txt->text[i]->NewCount = 0;
            for (int j = 0; j < txt->text[i]->CountSymbols; j++) {
                if (txt->text[i]->CountSymbols == txt->text[i]-
>MaxNewCount) {
                    txt->text[i]->MaxNewCount += ARRAY_SIZE;
                    txt->text[i]->ChangeSentence = realloc(txt-
>text[i]->ChangeSentence, txt->text[i]->MaxNewCount *

sizeof(wchar_t));
                }
                if (txt->text[i]->sentence[j] == '@') {
                    txt->text[i]->MaxNewCount += 4;
                    txt->text[i]->ChangeSentence = realloc(txt-
>text[i]->ChangeSentence, txt->text[i]->MaxNewCount *

sizeof(wchar_t));
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = '(';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = 'a';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = 't';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = ')';
                    txt->text[i]->NewCount++;
                } else if (txt->text[i]->sentence[j] == '#') {
                    txt->text[i]->MaxNewCount += 9;
                    txt->text[i]->ChangeSentence = realloc(txt-
>text[i]->ChangeSentence, txt->text[i]->MaxNewCount *

sizeof(wchar_t));
```

```
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = L'<';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = L'p';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = L'e';
                    txt->text[i]->NewCount++;
          txt->text[i]->ChangeSentence[txt->text[i]->NewCount] =
L'ш';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = L'ё';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = L'т';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = L'к';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = L'а';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = '>';
                    txt->text[i]->NewCount++;
                } else if (txt->text[i]->sentence[j] == '%') {
                    txt->text[i]->MaxNewCount += 9;
                    txt->text[i]->ChangeSentence = realloc(txt-
>text[i]->ChangeSentence, txt->text[i]->MaxNewCount *

sizeof(wchar_t));
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = '<';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = 'p';
                    txt->text[i]->NewCount++;
```

```
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = 'e';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = 'r';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = 's';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = 'e';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = 'n';
                    txt->text[i]->NewCount++;
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = 't';
                    txt->text[i]->NewCount++;
            txt->text[i]->ChangeSentence[txt->text[i]->NewCount] =
'>';
                    txt->text[i]->NewCount++;
                } else {
                    txt->text[i]->ChangeSentence[txt->text[i]-
>NewCount] = txt->text[i]->sentence[j];
                    txt->text[i]->NewCount++;
                }
             }
             txt->text[i]->ChangeSentence[txt->text[i]->NewCount] =
'\0';
             free(txt->text[i]->sentence);
             txt->text[i]->sentence = txt->text[i]->ChangeSentence;
             txt->text[i]->CountSymbols = txt->text[i]->NewCount;
        }
    }
}

int WordNumb(wchar_t* a){
    int Space = 0;
    int Numb = 0;
    for(int i = 0; i < wcslen(a); i++){
```

```
        if(Space > 0){
            break;
        }
        if (iswspace(a[i]) || a[i] == ','){
            Space++;
        } else{
            Numb += (int)a[i];
        }
    }
    return Numb;
}



int cmp(const void* a, const void* b) {
    if (WordNumb((*(Sentence **) a)->sentence) >
WordNumb((*(Sentence **) b)->sentence)) return 1;
    if (WordNumb((*(Sentence **) a)->sentence) ==
WordNumb((*(Sentence **) b)->sentence)) return 0;
    if (WordNumb((*(Sentence **) a)->sentence) <
WordNumb((*(Sentence **) b)->sentence)) return -1;

}
void sort(Text* txt) {
    qsort(txt->text, txt->CountSentence, sizeof(txt->text), cmp);
}

void del(Text* txt){
    int bool;
    int backbool;
    wchar_t consonant[] = {L'b', L'c', L'd', L'f', L'g', L'h',
L'k', L'j', L'l', L'm', L'n', L'p', L'q', L'r', L's', L't', L'v',
L'w', L'x', L'z', L'й' , L'ц', L'к', L'н', L'м', L'ш', L'щ', L'з',
L'х', L'ф', L'в', L'п', L'р', L'с', L'д', L'ж', L'ч', L'л', L'г',
L'б', L'т'};
    for(int i = 0; i < txt->CountSentence; i++){
        bool = 0;
        for(int j = txt->text[i]->CountSymbols - 2; j >= 0; j--){
            backbool = bool;
            for(int k = 0; k < wcslen(consonant); k++){
```

```
                if(txt->text[i]->sentence[j] == consonant[k]){
                    bool += 1;
                    break;
                }
            }
            if(backbool == bool){
                bool = 4;
            }
            if(bool == 3){
                free(txt->text[i]);
                txt->CountSentence--;
                for (int k = i; k < txt->CountSentence; k++){
                    txt->text[k] = txt->text[k+1];
                }
                i--;
                break;
            }
        }
    }
}

void CounterMin(Text* txt){
   int minute;
   int CountMinute = 0;
   int level;
   int TenToThePowerOf;
   for(int i = 0; i < txt->CountSentence; i++){
       for(int j = 0; j < wcslen(txt->text[i]->sentence) - 2; j++)
{
           if((j + 2) < wcslen(txt->text[i]->sentence) - 1){
               if ((txt->text[i]->sentence[j] == L's')
&& (txt->text[i]->sentence[j+1] == L'e')
&& (txt->text[i]->sentence[j+2] == L'c')
&& (txt->text[i]->sentence[j-1] == L' '))
            {
                  minute = 0;
                  level = 0;
                  for(int k = j - 2; (!iswspace(txt->text[i]-
>sentence[k]) && k >= 0); k--){
                      TenToThePowerOf = 1;
```

```
                        for(int p = 0; p < level; p++){
                            TenToThePowerOf *= 10;
                        }
                        minute += atoi((const char *) (txt-
>text[i]->sentence + k)) * TenToThePowerOf;
                        level++;
                    }
                    CountMinute += minute;
                }
            }
        }
    }
    wprintf(L"\t%d min\n", CountMinute / 60);
}


                continue;
            }
            wprintf(L">> Sentence num: %d || To current time: %d
minutes\n", i, abs(Today.tm_hour * 60 + Today.tm_min - Hour * 60 +
Minutes));
        }

}
```

название файла: IncludeAll.h:

```
#ifndef INC_INCLUDEALL_H
#define INC_INCLUDEALL_H

#include <stdio.h>
#include <stdlib.h>
#include <wchar.h>
#include <ctype.h>
#include <wctype.h>
#include <locale.h>
#include <math.h>

typedef struct Sentence Sentence;
typedef struct Text Text;
```

```
struct Sentence{
    int CountSymbols;
    int MaxCountSymbols;
    wchar_t* sentence;
    wchar_t* ChangeSentence;
    int NewCount;
    int MaxNewCount;
};

struct Text{
    Sentence** text;
    int CountSentence;
    int MaxCountSentence;
    wchar_t* word;
};

#endif
```

название файла: inputText.h:

```
#ifndef INPUTTEXT_H
#define INPUTTEXT_H

#include "IncludeAll.h"

void input(Text* Text);
void InputInStruct(Text* txt);
void output(Text* txt);
void CleanerSame(Text* txt);
#endif
```

название файла Funcs.h:

```
#ifndef FUNCS_H
#define FUNCS_H

#include "IncludeAll.h"

void change(Text* txt);
int WordNumb(wchar_t* a);
int cmp(const void* a, const void* b);
```

```
void sort(Text* txt);
void del(Text* txt);
void CounterMin(Text* txt);

#endif
```

название файла Makefile:

```
all: programm

programm: main.o Funcs.o inputText.o
     gcc main.o     Funcs.o inputText.o –o programm

main.o: main.c Funcs.o inputText.o
     gcc –c main.c –o main.o

inputText.o: inputText.c
     gcc –c inputText.c –o inputText.o

Funcs.o: Funcs.c
     gcc –c Funcs.c –o Funcs.o

clean:
     rm –rf *.o programm
```