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

Курсовая работа
по дисциплине «Программирование»
Тема: Обработка строк на языке Си

Студент гр. 0382 _____ Сергеев Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Сергеев Д.А.

Группа 0382

Тема работы: обработка строк на языке Си
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):

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

3. Заменить все символы ‘%’, ‘#’, ‘@’ на “<percent>”, “<решетка>”, “(at)” соответственно.
4. 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.

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: 02.11.2020
Дата сдачи реферата: 28.12.2020
Дата защиты реферата: 29.12.2020

Студент гр. 0382 _____ Сергеев Д.А.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В ходе выполнений курсовой работы была разработана программа для обработки текста на языке Си. Для хранения текста используются структуры и динамическое выделение памяти. Обработка и вывод текста реализована с помощью стандартных библиотек языка Си. На консоль выводится подсказки и контекстное меню для пользователя. При вводе некорректного значения будет выведена просьба ввести корректную команду. Для завершения работы с программой предусмотрена специальная команда. По завершению работы с текстом память, выделенная для него, освобождается. Для удобства сборки написан Makefile для утилиты make.

Пример работы программы приведён в приложении А.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной курсовой работе производится разработка стабильной консольной программы, производящей обработку введённого пользователем текста в соответствии с выбранной пользователем опцией обработки введённого текста.

Программа реализована при помощи использования структур (они используются для хранения данных о введённом тексте). Память под текст в программе выделяется динамически при помощи использования стандартных функций выделения памяти из стандартного заголовочного файла stdlib.h. Также разработанная программа имеет возможность обработки кириллических символов. Это возможность существует благодаря использованию функций из стандартных заголовочных файлов wchar.h и wctype.h, предназначенный для работы с широкими символами. Сборка программы реализуется при помощи утилиты Make и написанного файла Makefile.

Программа разработана для операционных систем на базе Linux. Разработка велась на операционной системе Ubuntu 20.04 при помощи интерактивной среды разработки CLion и текстового редактора Nano. Отладка программы производилась средствами интерактивной среды разработки и утилиты Valgrind

1. ЦЕЛИ И ДАДАЧИ

Цель работы: создать программу, которая способна выполнить считывание текста, его обработку в соответствии с выбранной опцией и вывод обработанного текста на консоль.

Задачи:

1. Реализовать считывание текста;
2. Реализовать контекстное меню, в котором пользователь сможет выбрать опцию обработки текста
3. Реализовать функции обработки текста
4. Реализовать вывод текста
5. Разбить программу на файлы, и написать Makefile для удобной сборки
6. Произвести отладку программы

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Структуры для считывания и хранения текста

```
struct Sentence
{
    wchar_t* sent;
};
typedef struct Sentence Sentence;

struct Text
{
    Sentence* doc;
    int text_size;
};
typedef struct Text Text;
```

Поля структуры Sentence:

- wchar_t* sent – указатель на начало предложения

Поля структуры Text:

- Sentence* doc – указатель на первый элемент массива предложений
- int text_size – количество предложений

2.2. Функции для считывания текста

Считывание текста происходит при помощи функций getText() и getSentence().

В функции Text getText() создаются экземпляры структур Text и Sentence, MyText и MySentence соотвественно. Далее до тех пор, пока возвращенное с помощью функции getSentence предложение не является символов переноса строки, в MySentence помещается возвращаемое значение функции getSentence, а далее структура MySentence помещается в массив MyText.doc под своим порядковым номером. В случае, если возвращенное с помощью функции предложение является символом переноса строки, то функция возвращает структуру MyText.

В функции Sentence getSentence() создаётся экземпляр структуры Sentence – ret_sent, далее с помощью функции wscanf происходит ввод предложения, если в начале предложения есть символы табуляции, то они не учитываются при вводе, символы, введёные с клавиатуры, записываются в MySentence.sent – полу структуры MySentence. После ввода точки пользователем функция возвращает структуру MySentence.

2.3 Функции для обработки текста

Text shift_text (Text test_text, int count, int* del) – удаляет из структуры test_text count предложений, индексы которых хранятся в массиве del.

Text DoSentencesUnique (Text FixText) – сначала находит индексы предложений, который уже встречались в тексте ранее (сравнение выполняется при помощи функции wcscasecmp()), помещает индексы в массив del, далее передаёт этот массив, его длину и FixText в функцию shift_text. Оставляет первое вхождение предложение. После проверки всех предложений возвращает структуру FixText.

Text UpSort(Text SortText) – функция, вызывает стандартную функцию qsort, в которую передаёт массив предложений SortText.doc, количество элементов в нём, размер каждого элемента и функцию-компаратор). Возвращает отсортированную структуру SortText.

int upsortcmp(const void* a, const void* b) – фукнция-компаратор, которая получает на вход указатели на указатели на объекты типа Sentence, после преобразования указателей, функция возвращает разность сумм ASCII sumb-suma.

Text ChangeSpecSym(Text RedText) – функция считает количество вхождений каждого из символов #, @, % в предложение, а запоминает индекс каждого из вхождений. Далее в фукнцию Shift_Sent() по очереди передаются предложения из текста RedText, массив с индексами вхождения каждого из символов, количество вхождений, уменьшенная на 1 длина фразы, которую

надо подставить вместо символа, и сам символ. После изменения всех предложений функция возвращает структуру RedText.

Sentence Shift_Sent(Sentence sc,int* beg,int count,int shift_size, wchar_t sym) – функция count раз увеличивает длину предложения на shift_size символов после чего, начиная с индексов из массива beg, меняет переданный в функцию символ на <английское написание символа>. Возвращает измененное предложение.

Text Delete_3_consonant_in_a_row(Text RedactText) – функция содержит два массива, содержащих заглавные согласные буквы из латинского и российского алфавита, функция доходит до последнего слова и проверяет посимвольно наличие вхождения подстроки из трёх подряд идущих согласных и сохраняет индексы(need_to_delete) и количество предложений (count) содержащих подстроку. После чего исполняется RedactText=shift_text(RedactText,count,need_to_delete) для того чтобы лишние предложения были удалены. И функция Delete_3_consonant_in_a_row() возвращает отредактированную структуру.

2.4 Функции вывода текста

void time (Text TimeText) – функция считает количество секунд, встречающихся в тексте и выводит на консоль это значение преобразованное к минутам. Это помогает осуществить функция wcsstr(TimeText.doc[i].sent,L" sec "), с помощью которой происходит поиск подстроки в строке, далее с помощью функции wcstol(digit,NULL,10), количество секунд преобразуется в число и складывается с суммой, полученной из предыдущих предложений.

void text_output(Text MyText) – выводит на консоль текст, начиная каждое предложение с новой строки.

2.5 Функция main

Для корректной работы с кириллическими символами в функции main используется функция setlocale().

Создаётся объект структуры Text MyText, далее выводится подсказка о вводе текст и о том, как ввод текста закончить. Ввод текста происходит с помощью функции getText(MyText), далее с помощью функции DoSentencesUnique() удаляются все повторные предложения. Функция main() возвращает значение функции Menu(MyText).

int Menu(Text MyText) – вызывает контекстное меню и ввод опции, в случае, если опция не входит в перечень доступных, то пользователю выводится просьба выбрать доступную опцию.

Опции:

- 0: Завершить программу, также освобождается память, выделенная под текст.
- 1: Посчитать и вывести в минутах количество секунд встречающихся в тексте. Количество секунд задается в виде подстроки “ <число> sec “. Реализовано с помощью функции time().
- 2: Отсортировать предложения по увеличению сумме кодов символов первого слова в предложении. Реализовано с помощью функции UpSort().
- 3: Заменить все символы ‘%’, ‘#’, ‘@’ на “<percent>”, “<решетка>”, “(at)” соответственно. Реализовано с помощью функции ChangeSpecSym().
- 4: Удалить все предложения которые заканчиваются на слово с тремя согласными подряд. Реализовано с помощью Delete_3_consonant_in_a_row().
- 5: Вывести текст на консоль. Реализовано с помощью text_output().

После выполнения каждой из функций происходит очистка использованной в функции памяти, вследствие чего не происходит утечек памяти.

2.6 Разделение программы на файлы и создание Makefile

Вся программа разделена на следующие файлы:

- main.c – основной файл программы, содержит функции main() и Menu(), с помощью директивы #include в этот файл включаются все заголовочные файлы.
- structures.h – заголовочный файл, содержит объявление структур.
- input_funcs.c – файл, содержит функции getSentence() и getText() с помощью директивы #include в этот файл включается файл input.h.
- input_funcs.h – файл, содержит объявление функций getSentence() и getText(), с помощью директивы #include в этот файл включаются стандартные библиотеки и файл structures.h.
- text_changing.c - файл, содержит функции shift_text(), DoSentencesUnique(), upsortcmp(), UpSort(), Shift_Sent(), ChangeSpecSym(), Delete_3_consonant_in_a_row(), с помощью директивы #include в этот файл включается файл text_changing.h.
- text_changing.h - файл, содержит объявление функций shift_text(), DoSentencesUnique(), upsortcmp(), UpSort(), Shift_Sent(), ChangeSpecSym(), Delete_3_consonant_in_a_row(), с помощью директивы #include в этот файл включаются стандартные библиотеки и файл structures.h.
- output_funcs.c - файл, содержит функции time() и text_output(), с помощью директивы #include в этот файл включается файл output_funcs.h.
- output_funcs.h - файл, содержит объявление функции time() и text_output(), с помощью директивы #include в этот файл включаются стандартные библиотеки и файл structures.h.
- Makefile – содержит команды для сборщика, прописаны все цели, команды и зависимости

ЗАКЛЮЧЕНИЕ

В результате данной курсовой работы были на практике применены приёмы работы со структурами, динамической память, стандартными функциями библиотек языка Си.

В ходе разработки программы была достигнута цель: разработана стабильная программа, способная запрашивать у пользователя нужную опцию обработки текста и выводить изменённый текст (исходный код программы см. в приложении Б).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

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

3. Cplusplus URL: http://cplusplus.com (дата обращения: 16.12.2020)

ПРИЛОЖЕНИЕ А
ПРИМЕР РАБОТЫ ПРОГРАММЫ

Вывод с консоли после запуска утилиты make:

```
dimasergeev02@dimasergeev02-VirtualBox:~/kursach/makefile$ make
gcc -c output_funcs.c
gcc -c text_changing.c
text_changing.c: In function ‘Shift_Sent’:
text_changing.c:85:18: warning: character constant too long for its type
   85 |         if (sym==L'%%')
      |                  ^~~~~
text_changing.c: In function ‘ChangeSpecSym’:
text_changing.c:114:41: warning: character constant too long for its type
  114 |             if (RedText.doc[i].sent[j]==L'%%')
      |                                         ^~~~~
text_changing.c:120:73: warning: character constant too long for its type
  120 | Text.doc[i]=Shift_Sent(RedText.doc[i], id_perc, count_perc,8,L'%%');
      |                                                             ^~~~~

gcc -c input_funcs.c
gcc -c main.c
gcc output_funcs.o text_changing.o input_funcs.o main.o -o menu
dimasergeev02@dimasergeev02-VirtualBox:~/kursach/makefile$
```

Консоль после запуска исполняемого файла menu:

```
dimasergeev02@dimasergeev02-VirtualBox:~/kursach/makefile$ ./menu
Введите текст(для окончания ввода нажмите Enter)
```

Вывод на консоли после ввода текста:

```
dimasergeev02@dimasergeev02-VirtualBox:~/kursach/makefile$ ./menu
Введите текст(для окончания ввода нажмите Enter)
Привет, как у тебя дела. ПриВЕТ, КАК у Тебя Дела. I was there 120 sec ago. Всё очень плхо. Я жду 24 апр
ля, чтобы пойти пог%л@ть#. ###############%%%%%@@#%@. Я гуляю по территории прка. Он не реагировал 12 se
c.
Введите число n, в зависимости от которого будут выполнены различные действия:
 0) Будет произведен выход из программы.
 1) Будет посчитано и выведено в минутах количество секунд встречающихся в тексте.
 2) Предложения будут отсортированы по увеличению сумме кодов символов первого слова в предложении.
 3) Все символы ‘%’, ‘#’, ‘@’ будут заменены на “<percent>”, “<решетка>”, “(at)” соответственно.
 4) Все предложения, которые заканчиваются на слово с тремя согласными подряд будут удалены.
 5) Будет произведен вывод текста на консоль.
Введите число n:5
Привет, как у тебя дела.
I was there 120 sec ago.
Всё очень плхо.
Я жду 24 апрля, чтобы пойти пог%л@ть#.
###############%%%%%@@#%@.
Я гуляю по территории прка.
Он не реагировал 12 sec.
```

Примеры выполнения каждой опции одна за другой:

```
Введите число n, в зависимости от которого будут выполнены различные действия:
 0) Будет произведен выход из программы.
 1) Будет посчитано и выведено в минутах количество секунд встречающихся в тексте.
 2) Предложения будут отсортированы по увеличению сумме кодов символов первого слова в предложении.
 3) Все символы ‘%’, ‘#’, ‘@’ будут заменены на “<percent>”, “<решетка>”, “(at)” соответственно.
 4) Все предложения, которые заканчиваются на слово с тремя согласными подряд будут удалены.
 5) Будет произведен вывод текста на консоль.
Введите число n:1
2 min 0 sec
```

```
Введите число n, в зависимости от которого будут выполнены различные действия:
 0) Будет произведен выход из программы.
 1) Будет посчитано и выведено в минутах количество секунд встречающихся в тексте.
 2) Предложения будут отсортированы по увеличению сумме кодов символов первого слова в предложении.
 3) Все символы ‘%’, ‘#’, ‘@’ будут заменены на “<percent>”, “<решетка>”, “(at)” соответственно.
 4) Все предложения, которые заканчиваются на слово с тремя согласными подряд будут удалены.
 5) Будет произведен вывод текста на консоль.
Введите число n:5
I was there 120 sec ago.
###############%%%%%@@#%@.
Я жду 24 апрля, чтобы пойти пог%л@ть#.
Я гуляю по территории прка.
Он не реагировал 12 sec.
Всё очень плхо.
Привет, как у тебя дела.
```

```
Введите число n, в зависимости от которого будут выполнены различные действия:
 0) Будет произведен выход из программы.
 1) Будет посчитано и выведено в минутах количество секунд встречающихся в тексте.
 2) Предложения будут отсортированы по увеличению сумме кодов символов первого слова в предложении.
 3) Все символы ‘%’, ‘#’, ‘@’ будут заменены на “<percent>”, “<решетка>”, “(at)” соответственно.
 4) Все предложения, которые заканчиваются на слово с тремя согласными подряд будут удалены.
 5) Будет произведен вывод текста на консоль.
Введите число n:3
Введите число n, в зависимости от которого будут выполнены различные действия:
 0) Будет произведен выход из программы.
 1) Будет посчитано и выведено в минутах количество секунд встречающихся в тексте.
 2) Предложения будут отсортированы по увеличению сумме кодов символов первого слова в предложении.
 3) Все символы ‘%’, ‘#’, ‘@’ будут заменены на “<percent>”, “<решетка>”, “(at)” соответственно.
 4) Все предложения, которые заканчиваются на слово с тремя согласными подряд будут удалены.
 5) Будет произведен вывод текста на консоль.
Введите число n:5
I was there 120 sec ago.
<решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><percent><percent><percent><percent><percent>(at)(at)<решетка><percent>(at).
Я жду 24 апрля, чтобы пойти пог<percent>л(at)ть<решетка>.
Я гуляю по территории прка.
Он не реагировал 12 sec.
Всё очень плхо.
Привет, как у тебя дела.
```

```
Введите число n, в зависимости от которого будут выполнены различные действия:
 0) Будет произведен выход из программы.
 1) Будет посчитано и выведено в минутах количество секунд встречающихся в тексте.
 2) Предложения будут отсортированы по увеличению сумме кодов символов первого слова в предложении.
 3) Все символы ‘%’, ‘#’, ‘@’ будут заменены на “<percent>”, “<решетка>”, “(at)” соответственно.
 4) Все предложения, которые заканчиваются на слово с тремя согласными подряд будут удалены.
 5) Будет произведен вывод текста на консоль.
Введите число n:4
Введите число n, в зависимости от которого будут выполнены различные действия:
 0) Будет произведен выход из программы.
 1) Будет посчитано и выведено в минутах количество секунд встречающихся в тексте.
 2) Предложения будут отсортированы по увеличению сумме кодов символов первого слова в предложении.
 3) Все символы ‘%’, ‘#’, ‘@’ будут заменены на “<percent>”, “<решетка>”, “(at)” соответственно.
 4) Все предложения, которые заканчиваются на слово с тремя согласными подряд будут удалены.
 5) Будет произведен вывод текста на консоль.
Введите число n:5
I was there 120 sec ago.
<решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><решетка><percent><percent><percent><percent><percent>(at)(at)<решетка><percent>(at).
Я жду 24 апрля, чтобы пойти пог<percent>л(at)ть<решетка>.
Он не реагировал 12 sec.
Привет, как у тебя дела.
```

```
Введите число n, в зависимости от которого будут выполнены различные действия:
 0) Будет произведен выход из программы.
 1) Будет посчитано и выведено в минутах количество секунд встречающихся в тексте.
 2) Предложения будут отсортированы по увеличению сумме кодов символов первого слова в предложении.
 3) Все символы ‘%’, ‘#’, ‘@’ будут заменены на “<percent>”, “<решетка>”, “(at)” соответственно.
 4) Все предложения, которые заканчиваются на слово с тремя согласными подряд будут удалены.
 5) Будет произведен вывод текста на консоль.
Введите число n:6
Вы ввели неправильное значение! Пожалуйста, выберите одну из предложенных опций:
Введите число n, в зависимости от которого будут выполнены различные действия:
 0) Будет произведен выход из программы.
 1) Будет посчитано и выведено в минутах количество секунд встречающихся в тексте.
 2) Предложения будут отсортированы по увеличению сумме кодов символов первого слова в предложении.
 3) Все символы ‘%’, ‘#’, ‘@’ будут заменены на “<percent>”, “<решетка>”, “(at)” соответственно.
 4) Все предложения, которые заканчиваются на слово с тремя согласными подряд будут удалены.
 5) Будет произведен вывод текста на консоль.
Введите число n:
```

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: main.c

```
#include "structures.h"
#include "text_changing.h"
#include "input_funcs.h"
#include "output_funcs.h"

#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>

int Menu(Text MyText);

int main() {
    setlocale(LC_ALL, "");
    Text MyText;
    wprintf(L"%s\n", "Введите текст(для окончания ввода нажмите
Enter)");
    MyText=get_Text();
    MyText=DoSentencesUnique(MyText);
    return (Menu(MyText));
}

int Menu(Text MyText)
{
    int n=3;
    while (1) {
        wprintf(L"Введите число n, в зависимости от которого
будут выполнены различные действия: \n 0) Будет произведен выход
из программы.\n 1) Будет посчитано и выведено в минутах
количество секунд встречающихся в тексте.\n 2) Предложения будут
отсортированы по увеличению сумме кодов символов первого слова в
```

```
предложении.\n 3) Все символы ‘%%’, ‘#’, ‘@’ будут заменены на “<percent>”, “<решетка>”, “(at)” соответственно.\n 4) Все предложения, которые заканчиваются на слово с тремя согласными подряд будут удалены.\n 5) Будет произведен вывод текста на консоль. \n");
        wprintf(L"Введите число n:");
        wscanf(L"%d",&n);
        switch (n) {
            case 0:
                for (int i = 0; i < MyText.text_size; i++) {
                    free(MyText.doc[i].sent);
                }
                free(MyText.doc);
                return 0;
            case 1:
                time(MyText);
                wprintf(L"\n");
                break;
            case 2:
                MyText = UpSort(MyText);
                break;
            case 3:
                MyText = ChangeSpecSym(MyText);
                break;
            case 4:
                MyText = Delete_3_consonant_in_a_row(MyText);
                break;
            case 5:
                text_output(MyText);
                break;
            default:
                wprintf(L"Вы ввели неправильное значение!
Пожалуйста, выберите одну из предложенных опций:\n");
                break;
        }
```

```
    }
                                    }
```

Файл structures.h:

```
#pragma once
#include <wchar.h>


struct Sentence{
    wchar_t* sent;
};
typedef struct Sentence Sentence;

struct Text{
    Sentence* doc;
    int text_size;
};
typedef struct Text Text;
```

Файл input_funcs.c:

```
#include "input_funcs.h"

Sentence get_sentence()
{
    Sentence ret_sent;
    ret_sent.sent=malloc(1*sizeof(wchar_t));
    wchar_t c='u';
    wscanf(L"%lc",&c);
    if (c!=L'\t' && c!=L'.' && c!=L' ')
    {
        if (c==L'\n')
        {
            ret_sent.sent[0]=c;
            return ret_sent;
        }
        int size=0;
```

```
        ret_sent.sent[size]=c;
        while (c!=L'.')
        {
            wscanf(L"%lc",&c);
            size+=1;

ret_sent.sent=realloc(ret_sent.sent,(size+1)*sizeof(wchar_t));
            ret_sent.sent[size]=c;
        }

ret_sent.sent=realloc(ret_sent.sent,(size+2)*sizeof(wchar_t));
        ret_sent.sent[size+1]=L'\0';
    }
    else
    {
        ret_sent.sent[0]=L'\0';
        return ret_sent;
    }
    return ret_sent;
}

Text get_Text()
{
    Text MyText;
    Sentence MySentence;
    MyText.text_size=0;
    MyText.doc=malloc((MyText.text_size+1)*(sizeof(Sentence
*)*2));
    while (1)
    {
        MySentence = get_sentence();
        if (MySentence.sent[0]!='\0' && MySentence.sent[0]!='\n')
        {
            MyText.doc[MyText.text_size] = MySentence;
            MyText.text_size += 1;
```

```
            MyText.doc = realloc(MyText.doc, (MyText.text_size +
1) * (sizeof(Sentence *)*2));
        }
        else
        {
            if (MySentence.sent[0]=='\n')
            {
                free (MySentence.sent);
                return MyText;
            }
        }
    }
}
```

Файл input_funcs.h:

```
#pragma once
#include "structures.h"
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>

Sentence get_sentence();

Text get_Text();
```

Файл text_changing.c:

```
#include "text_changing.h"

Text shift_text(Text test_text,int count, int* del)
{
    int point=0;
    for (int i=0;i<count;i++)
    {
        for (int j=del[point];j<test_text.text_size-1;j++)
```

```
        {
            test_text.doc[j]=test_text.doc[j+1];
        }
        point++;
        del[point]-=point;
        test_text.text_size-=1;

test_text.doc=realloc(test_text.doc,(test_text.text_size+1)*sizeo
f(Sentence*)*2);
    }
    free(del);
    return test_text;
}

Text DoSentencesUnique(Text FixText)
{
    for (int i=0;i<FixText.text_size;i++)
    {
        int*
need_to_delete=calloc(FixText.text_size,sizeof(int));
        int count=0;
        for (int j=i+1;j<FixText.text_size;j++)
        {
            int flag=1;
            if
((wcsncasecmp(FixText.doc[i].sent,FixText.doc[j].sent,wcslen(FixT
ext.doc[i].sent))))
            {
                flag=0;
            }
            if (flag==1)
            {
                need_to_delete[count]=j;
                count++;
            }
```

```
        }
        FixText=shift_text(FixText,count,need_to_delete);
    }
    return FixText;
}

int upsortcmp(const void* a, const void* b)
{
    int i=0;
    Sentence* aa=(Sentence*)a;
    Sentence* bb=(Sentence*)b;
    long int suma=0,sumb=0;
    while (aa->sent[i]!=L' ' && aa->sent[i]!=L'.' && aa-
>sent[i]!=L',')
    {
        suma+=aa->sent[i++];
    }
    i=0;
    while (bb->sent[i]!=L' ' && bb->sent[i]!=L'.' && bb-
>sent[i]!=L',')
    {
        sumb+=bb->sent[i++];
    }
    return suma-sumb;
}

Text UpSort(Text SortText)
{

qsort(SortText.doc,SortText.text_size,sizeof(Sentence),upsortcmp)
;
    return SortText;
}
```

```
Sentence Shift_Sent(Sentence sc,int* beg,int count,int
shift_size, wchar_t sym)
{
    int accumulate=0;
    for (int i=0;i<count;i++)
    {
        accumulate+=shift_size;
        for (int k=0;k<shift_size;k++)
        {

sc.sent=realloc(sc.sent,(wcslen(sc.sent)+2)*(sizeof(wchar_t)));
            int nach=wcslen(sc.sent);
            for (int j = nach; j > beg[i]; j--)
            {
                sc.sent[j]=sc.sent[j-1];
            }
            sc.sent[nach+1]='\0';
        }
        if (sym==L'%%')
        {
            sc.sent[beg[i]]=L'<'; sc.sent[beg[i]+1]=L'p';
sc.sent[beg[i]+2]=L'e'; sc.sent[beg[i]+3]=L'r';
sc.sent[beg[i]+4]=L'c'; sc.sent[beg[i]+5]=L'e';
sc.sent[beg[i]+6]=L'n'; sc.sent[beg[i]+7]=L't';
sc.sent[beg[i]+8]=L'>';
        }
        if (sym==L'#')
        {
            sc.sent[beg[i]]=L'<'; sc.sent[beg[i]+1]=L'p';
sc.sent[beg[i]+2]=L'e'; sc.sent[beg[i]+3]=L'ш';
sc.sent[beg[i]+4]=L'e'; sc.sent[beg[i]+5]=L'т';
sc.sent[beg[i]+6]=L'к'; sc.sent[beg[i]+7]=L'а';
sc.sent[beg[i]+8]=L'>';
        }
        if (sym==L'@')
```

```
        {
            sc.sent[beg[i]]=L'('; sc.sent[beg[i]+1]=L'a';
sc.sent[beg[i]+2]=L't'; sc.sent[beg[i]+3]=L')';
        }
        beg[i+1]+=accumulate;
    }
    return sc;
}

Text ChangeSpecSym(Text RedText)
{
    for (int i=0;i<RedText.text_size;i++)
    {
        int*
id_perc=calloc(wcslen(RedText.doc[i].sent),sizeof(int));
        int*
id_resh=calloc(wcslen(RedText.doc[i].sent),sizeof(int));
        int*
id_at=calloc(wcslen(RedText.doc[i].sent),sizeof(int));
        int count_perc=0;
        int count_resh=0;
        int count_at=0;
        for (int j=0;j<wcslen(RedText.doc[i].sent);j++)
        {
            if (RedText.doc[i].sent[j]==L'%%')
            {
                id_perc[count_perc]=j;
                count_perc++;
            }
        }
        RedText.doc[i]=Shift_Sent(RedText.doc[i], id_perc,
count_perc,8,L'%%');
        for (int j=0;j<wcslen(RedText.doc[i].sent);j++)
        {
            if (RedText.doc[i].sent[j]==L'#')
```

```
            {
                id_resh[count_resh]=j;
                count_resh++;
            }
        }
        RedText.doc[i]=Shift_Sent(RedText.doc[i], id_resh,
count_resh,8,L'#');
        for (int j=0;j<wcslen(RedText.doc[i].sent);j++)
        {
            if (RedText.doc[i].sent[j]==L'@')
            {
                id_at[count_at]=j;
                count_at++;
            }
        }
        RedText.doc[i]=Shift_Sent(RedText.doc[i], id_at,
count_at,3,L'@');
        free(id_at);
        free(id_perc);
        free(id_resh);
    }
    return RedText;
}

Text Delete_3_consonant_in_a_row(Text RedactText)
{
    wchar_t Lat_con[]=L"BCDFGHJKLMNPQRSTVWXYZ";
    wchar_t Rus_con[]=L"БВГДЖЗЙКЛМНПРСТФХЦЧШЩ";
    int* need_to_delete=calloc(1,sizeof(int));
    int count=0;
    for (int i=0;i<RedactText.text_size;i++)
    {
        int flag=0;
        int j=-1;
        wchar_t* beg=wcsstr(RedactText.doc[i].sent,L".");
```

```
        while (*(beg+j)!=' ' && (beg+j)!=RedactText.doc[i].sent)
        {
            j--;
        }
        int k;
        if ((beg+j)==RedactText.doc[i].sent)
        {
            k=0;
        }
        else
        {
            k = 1;
        }
        while (*(beg+j+k)!='.' && flag==0)
        {
            if (wcschr(Lat_con,towupper(*(beg+j+k)))!=0 ||
wcschr(Rus_con,towupper(*(beg+j+k)))!=0)
            {
                if (wcschr(Lat_con,towupper(*(beg+j+k+1)))!=0 ||
wcschr(Rus_con,towupper(*(beg+j+k+1)))!=0)
                {
                    if (wcschr(Lat_con,towupper(*(beg+j+k+2)))!=0
|| wcschr(Rus_con,towupper(*(beg+j+k+2)))!=0)
                    {
                        flag=1;
                        need_to_delete[count]=i;
                        count++;

need_to_delete=realloc(need_to_delete,(count+1)*sizeof(int));
                    }
                }
            }
            k++;
        }
    }
```

```
    RedactText=shift_text(RedactText,count,need_to_delete);
    return RedactText;
}
```

Файл text_changing.h:

```
#pragma once
#include "structures.h"
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>

Text shift_text(Text test_text,int count, int* del);

Text DoSentencesUnique(Text FixText);

int upsortcmp(const void* a, const void* b);

Text UpSort(Text SortText);

Sentence Shift_Sent(Sentence sc,int* beg,int count,int
shift_size, wchar_t sym);

Text ChangeSpecSym(Text RedText);

Text Delete_3_consonant_in_a_row(Text RedactText);
```

Файл output_funcs.c:

```
#include "output_funcs.h"

void time(Text TimeText)
{
    long int sum=0;
    for (int i=0;i<TimeText.text_size;i++)
    {
```

```
        wchar_t** replace=calloc(1,sizeof(wchar_t*));
        wchar_t* beg;
        int count=1;
        while (wcsstr(TimeText.doc[i].sent,L" sec ")!=0)
        {
            wchar_t* digit=calloc(1,sizeof(wchar_t));
            replace=realloc(replace,count*sizeof(wchar_t*));
            replace[count-1]=wcsstr(TimeText.doc[i].sent,L" sec
");
            count++;
            int j=-1;
            beg=wcsstr(TimeText.doc[i].sent,L" sec ");
            while (*(beg+j)!=' ' &&
(beg+j+1)!=TimeText.doc[i].sent)
            {
                j--;
            }
            int k=1;
            while (*(beg+j+k)!=' ')
            {
                digit=realloc(digit,(k+1)*sizeof(wchar_t));
                digit[k-1]=*(beg+j+k);
                k++;
            }
            digit=realloc(digit,(k+1)*sizeof(wchar_t));
            digit[k-1]='\0';
            sum=sum+wcstol(digit,NULL,10);
            *beg='1';
            free(digit);
        }
        for (int u=0;u<count-1;u++)
        {
            *(replace[u])=' ';
        }
        free(replace);
```

```
    }
    wprintf(L"%d min %d sec\n", sum/60, sum-(sum/60)*60);
}

void text_output(Text MyText)
{
for (int i = 0; i < MyText.text_size; i++)
     {
          wprintf(L"%ls\n", MyText.doc[i].sent);
        }
        wprintf(L"\n");
}
```

Файл output_funcs.h:

```
#pragma once
#include "structures.h"
#include <stdlib.h>
#include <locale.h>
#include <wchar.h>
#include <wctype.h>

void time(Text TimeText);
void text_output(Text MyText);
```

Файл Makefile:

```
all: output_funcs.o text_changing.o input_funcs.o main.o
     gcc output_funcs.o text_changing.o input_funcs.o main.o
main.o: output_funcs.h text_changing.h input_funcs.h structures.h
main.c
     gcc -c main.c
input_funcs.o: input_funcs.c input_funcs.h structures.h
     gcc -c input_funcs.c
text_changing.o: text_changing.c text_changing.h structures.h
     gcc -c text_changing.c
output_funcs.o: output_funcs.c output_funcs.h structures.h
```

```
	gcc -c output_funcs.c
clean:
	rm *.o
```