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

ОТЧЕТ

по курсовой работе

по дисциплине «Программирование»

Тема: «обработка текстовой информациию»

Студент гр. 1303 _____ Ахметгареев К.И

Преподаватель _____ Чайка К.В.

Санкт-Петербург, 2021

Цель работы.

Научиться работать со структурами, динамической памятью и функциями обработки строк и написать программу выполняющую обработку введенного пользователем текста произвольной длины.

Задание.

Вариант 1:

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) Вывести все предложения, которые являются анаграммами друг для друга. Учитывать надо только буквы и цифры.

2) отсортировать предложения (фактически, массив структур) по количеству заглавных букв в предложении.

3) Заменить каждую гласную буквы двумя другими буквами идущими следующими по алфавиту. Например, “ясЕнь” должно быть преобразовано в “абсЁЖнь”.

4) Заменить все вхождения одного слова (заданного пользователем) на другое слово (заданного пользователем).

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.

Выполнение работы.

Напишем структуру *struct Sentence.* Структура *Sentence* предназначена для хранения строки веутри структуры *Text.* Хранит единственное поле - *wchar_t *str*.

Следом напишем структуру *struct Text.* Структура Text предназначена для хранения массива структур Sentence.

Поля:

*Sentence *sen* - ранит массив структур Sentence.

size_t len - содержит реальное количество предложений.

Перейдём к основным функциям:

*int menu(Text *text)* - осуществляет управление обработкой текста пользователем. С помощью конструкции *switch* в зависимости от выбора пользователя вызывает одну из следующих четырёх функций. Функции обработки текста, вызывающиеся в этой функции и изменяющие текст принимают на вход указатель на структуру текст, поданный в *menu* в качестве параметра. Так же в меню предусмотрена возможность вывести обработанный текст на экран или выйти из программы.

void wait() - ожидает от пользователя нажатия на кнопку ENTER для продолжения работы программы.

void show_anagrams(Text text) – функция принимает на вход текст и выводит по группам все найденные предложения-анаграммы.

*void uppercase_sort(Text *text)* – функция принимает на вход указатель на текст и сортирует его по количеству гласных букв в предложениях.

*void vowels_to_next_two(Text *text)* – функция принимает на вход указатель на текст и заменяет каждую гласную букву в каждом предложении на две следующие за ней по алфавиту буквы.

*void swap(Text *text)* – функция принимает на вход указатель на текст и выводит сообщение-подсказку о том, что необходимо ввести два слова. Каждое вхождение в тексте первого слова будет заменено на второе. Функция не позволяет заменить не цельное слово и в случае ввода пользователем строки через пробел выдаёт сообщение об ошибке и возвращает пользователя в меню.

Функции работы с текстом:

Text text_input() – функция, посимвольно принимающая из стандартного потока ввода произвольного размера 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). Функция заносит входные данные в структуру *Text*. Функция выделяет на текст и отдельные предложения динамическую память.

*void text_free(Text *text)* – функция принимает на вход указатель на структуру *Text* и освобождает память, выделенную на структуру и её внутренние составляющие.

void print_text(Text text) – функция принимает на вход структуру *Text* и выводит текст на экран. Используется в функции *menu*.

*void text_cleaner(Text *text)* – функция принимает на вход указатель на структуру *Text*.

Функции необходимые для работы первой опции меню:

int is_anagram(Sentence sent1, Sentence sent2) – функция проверяет, являются ли содержимое полей str двух структур Sentence анаграммами. Возвращает 1 в случае, если являются, иначе 0.

*void clear_wstr(wchar_t *str)* – функция, очищающая строку от всех символов, не являющихся буквами или цифрами. Необходима для работы функции *is_anagram.*

*int wlexgraphic_cmp(const void *a, const void *b)* – функция-компаратор для функции *qsort*, необходимой для работы функции *is_anagram*, сравнивающая два символа по их расположению в таблице *ASCII*.

Функции необходимые для работы второй опции меню:

*void uppercase_sort(Text *text)* – функция, принимающая на вход структуру *Text* и сортирующая предложения в ней по количеству в них гласных букв. Для сортировки используется функция qsort из стандартной библиотеки.

*int wuppercase_cmp(const void *sen1, const void *sen2)* - функция-компаратор, необходимая для использования в функции *qsort* стандартной библиотеки. Сравнивает две широкие строки по возвращаемому функцией *count_uppers* значению.

*size_t count_uppers(wchar_t *str)* – функция принимает строку и возвращает количество гласных букв в ней.

Функции необходимые для работы третьей опции меню:

int is_vowel(wchar_t c) – функция проверяет, является ли символ гласной

буквой. Получает на вход символ, возвращает 1 в случае, если символ – гласная буква, иначе 0.

wchar_t get_next_char(wchar_t c) – функция принимает на вход широкий символ и возвращает следующий по алфавиту.

Функции необходимые для работы четвертой опции меню:

*int swap_in_string(wchar_t *str, wchar_t *prev, wchar_t *new)* – функция заменяет каждое слово *prev* в строке *str* на слово new. В следствие работы функции *is_sep_word* вхождениями слова в строку считаются лишь полноценные слова, а не часть слова.

*int is_sep_word(wchar_t *str, wchar_t *p, wchar_t *word)* – функция принимает указатель на строку, указатель на один из её элементов и строку, после чего проверяет, начинается ли в строке слово *new* с указателя *prev*

Для повышения читабельности кода и облегчения его поддержки с помощью директив *#define* было выведено большинство обращений к пользователю в отдельный файл, а с помощью директивы *#include* в программу были включены следующие заголовочные файлы стандартных библиотек языка Си:
stdio.h
stdlib.h
wctype.h
string.h
wchar.h
locale.h
Также в программу были включены заголовочные файлы *structures.h*, *functions.h* и *defines.h,* содержащие в себе создание структур, объявление всех используемых функций и все директивы *#define* соответственно.

MAKE-FILE, СБОРКА ПРОГРАММЫ:

Для облегчения поддержки кода программа была разбита на несколько файлов расширения «*.c», для упрощения сборки которых был написан Make-file.

Кроме целей, относящихся к непосредственной сборке программы, были добавлены следующие цели:
run – для объединения сборки программы и запуска исполняемого файла
clean – для очистки папки от объектных и исполняемого файлов.

```
CO = gcc -c
ALL = text.o option_1.o option_2.o option_3.o option_4.o
HEADERS = header/functions.h header/structures.h header/defines.h

all: main.o ${ALL}
	gcc main.o ${ALL} -o course_work

main.o: main.c ${HEADERS}
	${CO} main.c

text.o: text.c ${HEADERS}
	${CO} text.c

first_option.o: option_1.c ${HEADERS}
	${CO} first_option.c

second_option.o: option_1.c ${HEADERS}
	${CO} second_option.c

third_option.o: option_3.c ${HEADERS}
	${CO} third_option.c

fourth_option.o: option_4.c ${HEADERS}
	${CO} fourth_option.c


debug: main.o ${ALL}
	gcc -g main.o ${ALL} -o debugfile && gdb debugfile

run: all
	./course_work

clean:
	rm -rf *.o course_work debugfile
```

ИНСТРУКЦИЯ ПО ЗАПУСКУ

Чтобы запустить программ необходимо зайти через терминал в директорию *src*, после чего выполнить команду «make run». Далее следовать тому, что говорится в подсказках, выводящихся на экран.

ТЕСТИРОВАНИЕ

Option_1

```
make: *** No rule to make target 'gcc'.  Stop.
karim@karim-desktop:~/Documents/repo/pr-2021-1303/Akhmetgareyev_Karim_cw/src$ make run gcc main.o text.o option_1.o option_2.o option_3.o option_4.o -o course_work
gcc main.o text.o option_1.o option_2.o option_3.o option_4.o -o course_work
./course_work

Пожалуйста, введите текст|Please, enter the text:
        Клоун. КУЛОН. УКЛОН. RAT. Tar. Dusty. Study. Not anagram.

Выберите номер опции|Select the option number:
        1. Вывести все предложения-анаграммы|Output all anagram sentences
        2. Отсортировать предложения по количеству заглавных букв|Sort sentences by the number of uppercase letters
        3. Заменить все гласные буквы на две следующие за ней по алфавиту|Replace all vowel letters with the two following it alphabetically
        4. Заменить все слова А на слова Б|Replace all words A with words B
        5. Распечатать текст|Print the text
        0. Выйти|Exit
>>>1
Анаграммы (1):
        Клоун.
        КУЛОН.
        УКЛОН.
Анаграммы (2):
        RAT.
        Tar.
Анаграммы (3):
        Dusty.
        Study.

Для продолжения нажмите ENTER...|To continue, press ENTER...
```

Option_2

```
make: *** No rule to make target 'gcc'.  Stop.
karim@karim-desktop:~/Documents/repo/pr-2021-1303/Akhmetgareyev_Karim_cw/src$ make run gcc main.o text.o option_1.o option_2.o option_3.o option_4.o -o course_work
gcc main.o text.o option_1.o option_2.o option_3.o option_4.o -o course_work
./course_work

Пожалуйста, введите текст|Please, enter the text:
        ДОСТАТОЧНО ЗАГЛАВНЫХ БУКВ. Few UpperCase Letters. MORE UPPERcase букв.

Выберите номер опции|Select the option number:
        1. Вывести все предложения-анаграммы|Output all anagram sentences
        2. Отсортировать предложения по количеству заглавных букв|Sort sentences by the number of uppercase letters
        3. Заменить все гласные буквы на две следующие за ней по алфавиту|Replace all vowel letters with the two following it alphabetically
        4. Заменить все слова А на слова Б|Replace all words A with words B
        5. Распечатать текст|Print the text
        0. Выйти|Exit
>>>2

Текст отсортирован.|The text is sorted.

Для продолжения нажмите ENTER...|To continue, press ENTER...


Выберите номер опции|Select the option number:
        1. Вывести все предложения-анаграммы|Output all anagram sentences
        2. Отсортировать предложения по количеству заглавных букв|Sort sentences by the number of uppercase letters
        3. Заменить все гласные буквы на две следующие за ней по алфавиту|Replace all vowel letters with the two following it alphabetically
        4. Заменить все слова А на слова Б|Replace all words A with words B
        5. Распечатать текст|Print the text
        0. Выйти|Exit
>>>5
Few UpperCase Letters. MORE UPPERcase букв. ДОСТАТОЧНО ЗАГЛАВНЫХ БУКВ.

Для продолжения нажмите ENTER...|To continue, press ENTER...
```

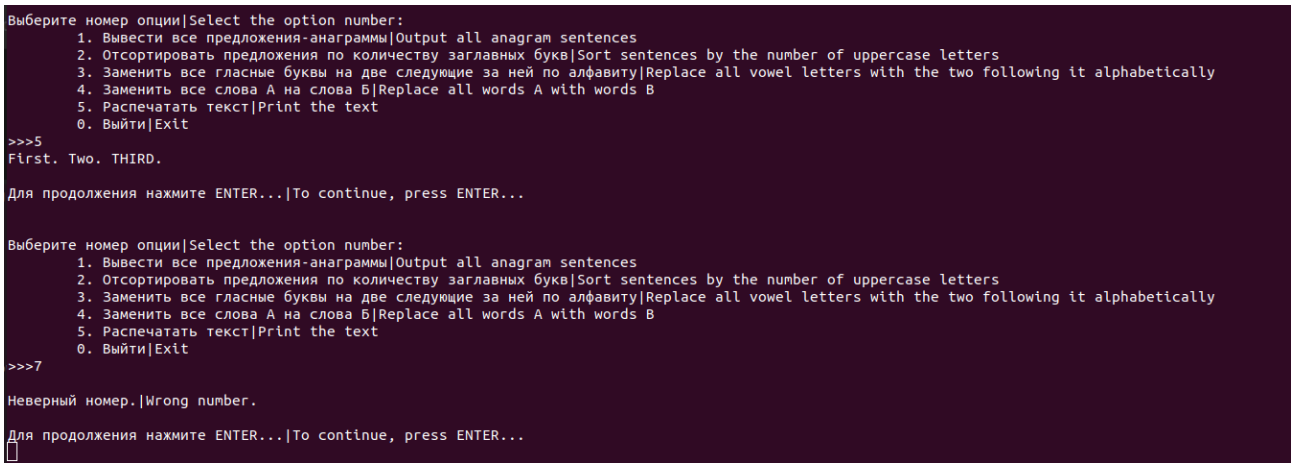
Option_3

```
karim@karim-desktop:~/Documents/repo/pr-2021-1303/Akhmetgareyev_Karim_cw/src$ make run gcc main.o text.o option_1.o option_2.o option_3.o option_4.o -o course_work
gcc main.o text.o option_1.o option_2.o option_3.o option_4.o -o course_work
./course_work

Пожалуйста, введите текст|Please, enter the text:
        ДОСТАТОЧНО ЗАГЛАВНЫХ БУКВ. Few UpperCase Letters. MORE UPPERcase букв.

Выберите номер опции|Select the option number:
        1. Вывести все предложения-анаграммы|Output all anagram sentences
        2. Отсортировать предложения по количеству заглавных букв|Sort sentences by the number of uppercase letters
        3. Заменить все гласные буквы на две следующие за ней по алфавиту|Replace all vowel letters with the two following it alphabetically
        4. Заменить все слова А на слова Б|Replace all words A with words B
        5. Распечатать текст|Print the text
        0. Выйти|Exit
>>>3

Гласные успешно заменены|Vowels have been successfully replaced

Для продолжения нажмите ENTER...|To continue, press ENTER...


Выберите номер опции|Select the option number:
        1. Вывести все предложения-анаграммы|Output all anagram sentences
        2. Отсортировать предложения по количеству заглавных букв|Sort sentences by the number of uppercase letters
        3. Заменить все гласные буквы на две следующие за ней по алфавиту|Replace all vowel letters with the two following it alphabetically
        4. Заменить все слова А на слова Б|Replace all words A with words B
        5. Распечатать текст|Print the text
        0. Выйти|Exit
>>>5
ДПРСТБВТПРЧНПР ЗБВГЛБВВНЬЭХ БФХКВ. Ffgw VWppfgrCbcsfg Lfgttfgrs. MPQRFG VWPPFGRcbcsfg бфхкв.

Для продолжения нажмите ENTER...|To continue, press ENTER...
```

Option_4

```
karim@karim-desktop:~/Documents/repo/pr-2021-1303/Akhmetgareyev_Karim_cw/src$ make run gcc main.o text.o option_1.o option_2.o option_3.o option_4.o -o course_work
gcc main.o text.o option_1.o option_2.o option_3.o option_4.o -o course_work
./course_work

Пожалуйста, введите текст|Please, enter the text:
        First. Second. THIRD.

Выберите номер опции|Select the option number:
        1. Вывести все предложения-анаграммы|Output all anagram sentences
        2. Отсортировать предложения по количеству заглавных букв|Sort sentences by the number of uppercase letters
        3. Заменить все гласные буквы на две следующие за ней по алфавиту|Replace all vowel letters with the two following it alphabetically
        4. Заменить все слова А на слова Б|Replace all words A with words B
        5. Распечатать текст|Print the text
        0. Выйти|Exit
>>>4
Введите слово, которое подлежит замене|Enter the word to be replaced: Second
Введите слово, на которое нужно заменить|Enter the word you want to replace: Two

Слова успешно заменены|Words have been successfully replaced

Для продолжения нажмите ENTER...|To continue, press ENTER...


Выберите номер опции|Select the option number:
        1. Вывести все предложения-анаграммы|Output all anagram sentences
        2. Отсортировать предложения по количеству заглавных букв|Sort sentences by the number of uppercase letters
        3. Заменить все гласные буквы на две следующие за ней по алфавиту|Replace all vowel letters with the two following it alphabetically
        4. Заменить все слова А на слова Б|Replace all words A with words B
        5. Распечатать текст|Print the text
        0. Выйти|Exit
>>>5
First. Two. THIRD.

Для продолжения нажмите ENTER...|To continue, press ENTER...
```

Обработка ошибок



Выводы.

В ходе работы были изучены структуры, работа с динамической памятью и функциями обработки строк и написана программа, выполняющая разнообразную обработку получаемого на вход текста.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#include "header/structures.h"
#include "header/functions.h"
#include "header/defines.h"



int main()
{
setlocale(LC_CTYPE, "");

wprintf(GREETING);
Text text = text_input();
while (!text.len) {
	wprintf(INPUT_ERROR);
	wait();
	wprintf(GREETING);
	text = text_input();
}
text_cleaner(&text);

while(menu(&text));

text_free(&text);
```

```
return 0;
}

// Меню
int menu(Text *text)
{
wprintf(HELP_MESSAGE);
int choice;
choice = (int)getwchar() - (int)L'0';
if (getwchar() != L'\n'){
        while(getwchar() != L'\n');
        choice = -1;
}
switch (choice)
{
case 1:
        show_anagrams(*text);
        wait();
        CONTINUE;
case 2:
        uppercase_sort(text);
        wait();
        CONTINUE;
case 3:
        vowels_to_next_two(text);
        wait();
        CONTINUE;
case 4:
        swap(text);
        wait();
        CONTINUE;
case 5:
        print_text(*text);
        wait();
        CONTINUE;
```

```
case 0:
        wprintf(L"\nВыход из программы!|Exiting the program!\n");
        EXIT;
default:
        wprintf(L"\nНеверный номер.|Wrong number.\n");
        wait();
        CONTINUE;
}
}


void wait()
{
wprintf(WAITING_MESSAGE);
while(getwchar() != '\n');
}
```

Названиe файла: text.c

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

#include "header/structures.h"
#include "header/functions.h"
#include "header/defines.h"

Text text_input()
{
Text text = {NULL, 0};
size_t sen_size = 1;
size_t str_size = INIT_STR_SIZE;
```

```
	size_t i = 0, j = 0;
	wchar_t c;

	text.sen = (Sentence *)calloc(sen_size, sizeof(Sentence));
	text.sen[i].str = (wchar_t *)calloc(str_size, sizeof(wchar_t));
	while (1) {
		if (j == str_size - 1) {
			str_size *= 2;
			text.sen[i].str = (wchar_t *)realloc(text.sen[i].str, str_size * sizeof(wchar_t));
		}
		c = getwchar();
		if (c == L'.') {
			if (i == sen_size - 1) {
				sen_size *= 2;
				text.sen = (Sentence *)realloc(text.sen, sen_size * sizeof(Sentence));
			}
			text.sen[i].str[j++] = c;
			text.sen[i].str[j] = L'\0';
			i++; j = 0;
			str_size = INIT_STR_SIZE;
			text.sen[i].str = (wchar_t *)calloc(str_size, sizeof(wchar_t));
			while (iswspace(c = getwchar()) && c != L'\n');
		}
		if (c == L'\n') break;
		text.sen[i].str[j++] = c;
	}
	if (i != 0)
		text.len = i;
```

```
else return (Text){NULL, 0};
return text;
}


void text_free(Text *text)
{
for (size_t i = 0; i < text->len; i++) {
        free(text->sen[i].str);
}
free(text->sen);
}


void print_text(Text text)
{
for (size_t i = 0; i < text.len; i++) {
        wprintf(L"%ls ", text.sen[i].str);
}
wprintf(L"\n");
}


void text_cleaner(Text *text)
{
for (size_t i = 0; i < text->len; i++)
        for (size_t j = i + 1; j < text->len; j++)
                if (!wcscasecmp(text->sen[i].str, text->sen[j].str)) {
                        for (size_t k = j; k < text->len - 1; k++)
                                text->sen[k].str = text->sen[k+1].str;
                        text->len--;
```

```
                        j--;
                }
        }
```

Название файла: option_1.c

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

#include "header/structures.h"
#include "header/functions.h"
#include "header/defines.h"



void show_anagrams(Text text)
{
        size_t *anagram_flags = (size_t*)calloc(text.len, sizeof(size_t));

        size_t max_flag = 0;
        for (size_t i = 0; i < text.len; i++) {
                if (anagram_flags[i]) continue;
                for (size_t j = i + 1; j < text.len; j++)
                        if (is_anagram(text.sen[i], text.sen[j])) {
                                if (!anagram_flags[i]) max_flag++;
                                anagram_flags[i] = max_flag;
                                anagram_flags[j] = max_flag;
                        }
        }
        if (!max_flag) wprintf(ANAGRAM_ERROR);
        size_t k = 0;
        while (k++ != max_flag) {
                wprintf(L"Анаграммы (%d):\n", k);
                for (size_t i = 0; i < text.len; i++)
                        if (anagram_flags[i] == k)
```

```
                wprintf(L"\t%ls\n", text.sen[i].str);
        }
        free(anagram_flags);
}


int is_anagram(Sentence sent_1, Sentence sent_2)
{
        wchar_t *new_str_1 = (wchar_t *)malloc((wcslen(sent_1.str)+1) * sizeof(wchar_t));
        wcscpy(new_str_1, sent_1.str);
        clear_wstr(new_str_1);

        wchar_t *new_str_2 = (wchar_t *)malloc((wcslen(sent_2.str)+1) * sizeof(wchar_t));
        wcscpy(new_str_2, sent_2.str);
        clear_wstr(new_str_2);

        if (wcslen(new_str_1) != wcslen(new_str_2)) {
                free(new_str_1); free(new_str_2);
                return 0;
   }
        qsort(new_str_1, wcslen(new_str_1), sizeof(wchar_t), wlexgraphic_cmp);
        qsort(new_str_2, wcslen(new_str_2), sizeof(wchar_t), wlexgraphic_cmp);
        int result = !wcscmp(new_str_1, new_str_2);
        free(new_str_1); free(new_str_2);
        return result;
}


void clear_wstr(wchar_t *str)
{
        size_t k = 0;
        for (size_t i = 0; i < wcslen(str); i++) {
                if (iswalnum(str[i])) {
                        str[k++] = towupper(str[i]);
                }
```

```
        }
        str[k] = L'\0';
}

int wlexgraphic_cmp(const void *a, const void *b)
{
        wchar_t aa = *(wchar_t *)a;
        wchar_t bb = *(wchar_t *)b;
        if (aa <  bb) return -1;
        if (aa == bb) return 0;
        if (aa >  bb) return 1;
        return -1;
}
```

Названиe файла: option_2.c

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

#include "header/structures.h"
#include "header/functions.h"
#include "header/defines.h"


void uppercase_sort(Text *text)
{
        qsort(text->sen, text->len, sizeof(Sentence), wuppercase_cmp);
        wprintf(SORT_SUCCESS);
}


int wuppercase_cmp(const void *sen1, const void *sen2)
{
        wchar_t *string1 = (*(Sentence *)sen1).str;
        wchar_t *string2 = (*(Sentence *)sen2).str;
        size_t uprs1 = count_uppers(string1);
        size_t uprs2 = count_uppers(string2);
        if (uprs1 <  uprs2) return -1;
        if (uprs1 == uprs2) return 0;
        if (uprs1 >  uprs2) return 1;
        return -1;
```

```
}


size_t count_uppers(wchar_t *str)
{
        size_t res = 0;
        for (size_t i = 0; i < wcslen(str); i++) {
                if (iswupper(str[i])) res++;
        }
        return res;
}
```

Название файла: option_3.c

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

#include "header/structures.h"
#include "header/functions.h"
#include "header/defines.h"



void vowels_to_next_two(Text *text)
{
   int flag = 0;
        for (size_t i = 0; i < text->len; i++) {
                size_t needed_memory = 1 + 2 * wcslen(text->sen[i].str) * sizeof(wchar_t);
                text->sen[i].str = (wchar_t *)realloc(text->sen[i].str, needed_memory);
                for (size_t j = 0; j < wcslen(text->sen[i].str); j++) {
                        if (is_vowel(text->sen[i].str[j])) {
           flag = 1;
                                memmove(&text->sen[i].str[j+1],
                                        &text->sen[i].str[j],
                                        sizeof(wchar_t) * (wcslen(text->sen[i].str) + 1 - j));
                                text->sen[i].str[j] = get_next_char(text->sen[i].str[j]);
                                text->sen[i].str[j+1] = get_next_char(text->sen[i].str[j]);
```

```
                                j++;
                        }
                }
        }
        if (!flag) wprintf(VOWELS_ERROR);
        else wprintf(VOWELS_SUCCESS);
}


int is_vowel(wchar_t c)
{
        return wcschr(VOWELS, towupper(c)) != NULL;
}


wchar_t get_next_char(wchar_t c) {
        if (c == L'Я') return L'А';
        if (c == L'я') return L'а';
        if (c == L'E') return L'Ё';
        if (c == L'e') return L'ё';
        if (c == L'Ё') return L'Ж';
        if (c == L'ё') return L'ж';
        if (c == L'Z') return L'A';
        if (c == L'z') return L'a';
        return iswalpha ? c + 1 : c;
}
```

Название файла: option_4.c

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

```
#include "header/structures.h"
#include "header/functions.h"
#include "header/defines.h"



void swap(Text *text)
{
	int flag = 0;
	wchar_t prev[MAX_WORD_LEN], new[MAX_WORD_LEN];

	wprintf(PREV_WORD_PLEASE);
	fgetws(prev, 200, stdin);
	prev[wcslen(prev) - 1] = L'\0';
	for (size_t i = 0; i < wcslen(prev); i++)
		if (!iswalpha(prev[i])) {
			wprintf(INPUT_ERROR);
			flag = 1;
			return;
		}

	wprintf(NEW_WORD_PLEASE);
	fgetws(new, 200, stdin);
	new[wcslen(new) - 1] = L'\0';
	for (size_t i = 0; i < wcslen(new); i++)
		if (!iswalpha(new[i])) {
			wprintf(INPUT_ERROR);
			flag = 1;
			return;
		}

	if (!flag) {
		for (size_t i = 0; i < text->len; i++) {
			if (swap_in_string(text->sen[i].str, prev, new)) flag = 1;
		}
```

```
                if (!flag) wprintf(SWAP_ERROR);
        else wprintf(SWAP_SUCCESS);
          }
}


int swap_in_string(wchar_t *str, wchar_t *prev, wchar_t *new)
{
   int flag = 0;
   size_t prev_len = wcslen(prev), new_len = wcslen(new);
   for (size_t i = 0; str[i + prev_len - 1] != L'\0'; i++) {
      if (is_sep_word(str, str+i, prev)) {
         flag = 1;
         if (new_len > prev_len) {
            str = (wchar_t *)realloc(str, (wcslen(str) + new_len * 10) * sizeof(wchar_t));
            memmove(str + i + new_len, str + i, (wcslen(str) - i + 2) * sizeof(wchar_t));
         }
         else {
                memmove(str + i + new_len, str + i + prev_len, (wcslen(str) + 1 - i - prev_len) *
sizeof(wchar_t));
         }
         wcsncpy(str + i, new, new_len);
      }
   }
   return flag;
}


int is_sep_word(wchar_t *str, wchar_t *p, wchar_t *word)
{
   size_t len = wcslen(word);
   return ((!wcsncasecmp(word, p, len)) && ((p == str && wcschr(L" ,.", *(str + len))) ||
                        wcschr(L" ,.\0", *(p + len)) && wcschr(L" ,", *(p - 1))));
}
```

Название файла: defines.h

```
#define CONTINUE return 1;
#define EXIT return 0;

#define VOWELS L"AEIOUYАЕЁИОУЫЭЮЯ"
#define MAX_WORD_LEN 200
#define INIT_STR_SIZE 32

#define WAITING_MESSAGE L"\nДля продолжения нажмите ENTER...|To continue, press ENTER...\n"
#define GREETING L"\nПожалуйста, введите текст|Please, enter the text:\n\t"
#define PREV_WORD_PLEASE L"Введите слово, которое подлежит замене|Enter the word to be replaced: "
#define NEW_WORD_PLEASE L"Введите слово, на которое нужно заменить|Enter the word you want to replace: "
#define HELP_MESSAGE L"\nВыберите номер опции|Select the option number:\n\t1. Вывести все предложения-анаграммы|Output all anagram sentences\n\t2. Отсортировать предложения по количеству заглавных букв|Sort sentences by the number of uppercase letters\n\t3. Заменить все гласные буквы на две следующие за ней по алфавиту|Replace all vowel letters with the two following it alphabetically\n\t4. Заменить все слова А на слова Б|Replace all words A with words B\n\t5. Распечатать текст|Print the text\n\t0. Выйти|Exit\n>>>"

#define ANAGRAM_ERROR L"\nАнаграмм не найдено.|No anagrams found.\n"
#define SWAP_ERROR L"\nТекст не был изменён.|The text has not been changed.\n"
#define INPUT_ERROR L"\nВвод некорректен.|The input is incorrect.\n"
#define VOWELS_ERROR L"\nВ тексте нет гласных букв.|There are no vowel letters in the text."
```

#define SORT_SUCCESS L"\nТекст отсортирован.|The text is sorted.\n"

#define SWAP_SUCCESS L"\nСлова успешно заменены|Words have been successfully replaced\n"

#define VOWELS_SUCCESS L"\nГласные успешно заменены|Vowels have been successfully replaced\n"

Названиe файла: functions.h

*int menu(Text *);*

void show_anagrams(Text);

*void uppercase_sort(Text *);*

*void vowels_to_next_two(Text *);*

*void swap(Text *);*

Text text_input();

void print_text(Text);

*void text_free(Text *);*

*void text_cleaner(Text *);*

void wait();

int is_anagram(Sentence, Sentence);

*void clear_wstr(wchar_t *);*

*size_t count_uppers(wchar_t *);*

wchar_t get_next_char(wchar_t);

int is_vowel(wchar_t);

*int is_sep_word(wchar_t *str, wchar_t *p, wchar_t *word);*

*int swap_in_string(wchar_t *str, wchar_t *prev, wchar_t *new);*

*int wuppercase_cmp(const void *, const void *);*

*int wlexgraphic_cmp(const void *, const void *);*

Название файла: structures.h

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

typedef struct {
    Sentence *sen;
    size_t len;
} Text;
```