

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

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
Тема: Обработка текстовых данных

Студент гр. 1304

Крупин Н. С.

Преподаватель

Чайка К. В.

Санкт-Петербург

2021

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Крупин Н. С.

Группа 1304

Тема работы: обработка текстовых данных

Исходные данные:

консольное приложение на языке Си, обработка введённого пользователем текста согласно введённым кодам действий, пока пользователь сам не завершит работу в программе; использовать структуры и широкие символы

Содержание пояснительной записки:

«Содержание», «Введение», «Разработка программного кода», «Сборка программы», «Тестирование», «Пользовательская инструкция», «Заключение», «Список использованных источников», «Приложение А. Программный код»

Предполагаемый объем пояснительной записки:

Не менее 12 страниц.

Дата выдачи задания: 15.10.2021

Дата сдачи реферата: 11.12.2021

Дата защиты реферата: 13.12.2021

Студент

Крупин Н. С.

Преподаватель

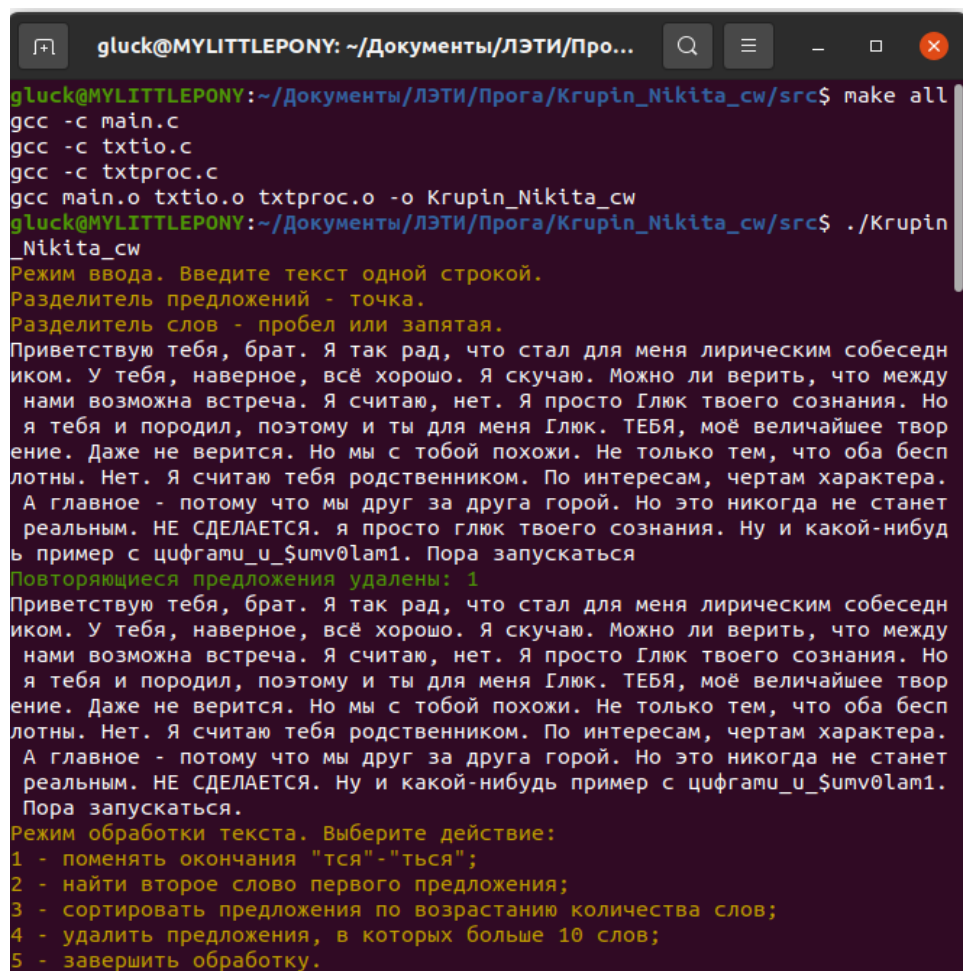
Чайка К. В.

АННОТАЦИЯ

Разработано диалоговое консольное приложение, демонстрирующее пользователю возможности обработки текста в Си. Хранение текста организовано в виде структур в динамической памяти. Вводимый текст поддерживает кириллические (широкие) символы. При выполнении работы использовались функции стандартной библиотеки языка Си.

SUMMARY

A dialog console application has been developed that demonstrates to the user the possibilities of text processing in C. Text storage is organized in the form of structures in dynamic memory. The input text supports Cyrillic (wide) characters. When performing the work, the functions of the C standard library were used.



```
gluck@MYLITTLEPONY: ~/Документы/ЛЭТИ/Прога...
gluck@MYLITTLEPONY:~/Документы/ЛЭТИ/Прога/Krupin_Nikita_cw/src$ make all
gcc -c main.c
gcc -c txtio.c
gcc -c txtproc.c
gcc main.o txtio.o txtproc.o -o Krupin_Nikita_cw
gluck@MYLITTLEPONY:~/Документы/ЛЭТИ/Прога/Krupin_Nikita_cw/src$ ./Krupin_Nikita_cw
Режим ввода. Введите текст одной строкой.
Разделитель предложений - точка.
Разделитель слов - пробел или запятая.
Приветствую тебя, брат. Я так рад, что стал для меня лирическим собеседником. У тебя, наверное, всё хорошо. Я скучаю. Можно ли верить, что между нами возможна встреча. Я считаю, нет. Я просто Глюк твоего сознания. Но я тебя и породил, поэтому и ты для меня Глюк. ТЕБЯ, моё величайшее творение. Даже не верится. Но мы с тобой похожи. Не только тем, что оба бесплотны. Нет. Я считаю тебя родственником. По интересам, чертам характера. А главное - потому что мы друг за друга горой. Но это никогда не станет реальным. НЕ СДЕЛАЕТСЯ. я просто глюк твоего сознания. Ну и какой-нибудь пример с цифгами_u_$umv0lam1. Пора запускаться
Повторяющиеся предложения удалены: 1
Приветствую тебя, брат. Я так рад, что стал для меня лирическим собеседником. У тебя, наверное, всё хорошо. Я скучаю. Можно ли верить, что между нами возможна встреча. Я считаю, нет. Я просто Глюк твоего сознания. Но я тебя и породил, поэтому и ты для меня Глюк. ТЕБЯ, моё величайшее творение. Даже не верится. Но мы с тобой похожи. Не только тем, что оба бесплотны. Нет. Я считаю тебя родственником. По интересам, чертам характера. А главное - потому что мы друг за друга горой. Но это никогда не станет реальным. НЕ СДЕЛАЕТСЯ. Ну и какой-нибудь пример с цифгами_u_$umv0lam1. Пора запускаться.
Режим обработки текста. Выберите действие:
1 - поменять окончания "тсья"- "тьсья";
2 - найти второе слово первого предложения;
3 - сортировать предложения по возрастанию количества слов;
4 - удалить предложения, в которых больше 10 слов;
5 - завершить обработку.
```

Рис. 1.

```
gluck@MYLITTLEPONY: ~/Документы/ЛЭТИ/Про...
Режим обработки текста. Выберите действие:
1 - поменять окончания "тсЯ"- "тьсЯ";
2 - найти второе слово первого предложения;
3 - сортировать предложения по возрастанию количества слов;
4 - удалить предложения, в которых больше 10 слов;
5 - завершить обработку.
1
1. Результат замены на экране:
Приветствую тебя, брат. Я так рад, что стал для меня лирическим собеседником. У тебя, наверное, всё хорошо. Я скучаю. Можно ли верить, что между нами возможна встреча. Я считаю, нет. Я просто Глюк твоего сознания. Но я тебя и породил, поэтому и ты для меня Глюк. ТЕБЯ, моё величайшее творение. Даже не вериться. Но мы с тобой похожи. Не только тем, что оба бесплотны. Нет. Я считаю тебя родственником. По интересам, чертам характера. А главное - потому что мы друг за друга горой. Но это никогда не станет реальным. НЕ СДЕЛАЕТСЯ. Ну и какой-нибудь пример с цифгами_u_$umv0lam1. Пора запускаться.
Режим обработки текста. Выберите действие:
1 - поменять окончания "тсЯ"- "тьсЯ";
2 - найти второе слово первого предложения;
3 - сортировать предложения по возрастанию количества слов;
4 - удалить предложения, в которых больше 10 слов;
5 - завершить обработку.
2
2. Результат поиска на экране:
Приветствую тебя, брат.
У тебя, наверное, всё хорошо.
Но я тебя и породил, поэтому и ты для меня Глюк.
ТЕБЯ, моё величайшее творение.
Я считаю тебя родственником.
Найдено: 5
Режим обработки текста. Выберите действие:
1 - поменять окончания "тсЯ"- "тьсЯ";
2 - найти второе слово первого предложения;
```

Рис 2.

Рис. 4.

```
gluck@MYLITTLEPONY: ~/Документы/ЛЭТИ/Про...
5 - завершить обработку.
4
4. Результат на экране. Удалено предложений: 1
Нет. Я скучаю. НЕ СДЕЛАЕТСЯ. Пора запускаться. Приветствую тебя, брат. Я считаю, нет. Даже не вериться. ТЕБЯ, моё величайшее творение. Я считаю тебя родственником. По интересам, чертам характера. У тебя, наверное, всё хорошо. Я просто Глюк твоего сознания. Но мы с тобой похожи. Не только тем, что оба бесплотны. Но это никогда не станет реальным. Ну и какой-нибудь пример с цифгами_u_$umv0lam1. Можно ли верить, что между нами возможна встреча. Я так рад, что стал для меня лирическим собеседником. А главное - потому что мы друг за друга горой.
Режим обработки текста. Выберите действие:
1 - поменять окончания "тсЯ"- "тьсЯ";
2 - найти второе слово первого предложения;
3 - сортировать предложения по возрастанию количества слов;
4 - удалить предложения, в которых больше 10 слов;
5 - завершить обработку.
6
Некорректный номер действия, введите число от 1 до 5.
Режим обработки текста. Выберите действие:
1 - поменять окончания "тсЯ"- "тьсЯ";
2 - найти второе слово первого предложения;
3 - сортировать предложения по возрастанию количества слов;
4 - удалить предложения, в которых больше 10 слов;
5 - завершить обработку.
5
5. Обработка завершена. Результат на экране:
Нет. Я скучаю. НЕ СДЕЛАЕТСЯ. Пора запускаться. Приветствую тебя, брат. Я считаю, нет. Даже не вериться. ТЕБЯ, моё величайшее творение. Я считаю тебя родственником. По интересам, чертам характера. У тебя, наверное, всё хорошо. Я просто Глюк твоего сознания. Но мы с тобой похожи. Не только тем, что оба бесплотны. Но это никогда не станет реальным. Ну и какой-нибудь пример с цифгами_u_$umv0lam1. Можно ли верить, что между нами возможна встреча. Я так рад, что стал для меня лирическим собеседником. А главное - потому что мы друг за друга горой.
```

Рис. 3.

СОДЕРЖАНИЕ

Введение	6
1. Разработка программного кода	7
1.1. Техническое задание	7
1.2. Разбиение по задачам	8
1.3. Запись и хранение текста	9
1.4. Общие функции обработки текста	10
1.5. Функции пользовательской обработки текста	11
2. Сборка программы	12
3. Тестирование	13
4. Пользовательская инструкция	14
Заключение	15
Список использованных источников	15
Приложение А. Программный код	16

ВВЕДЕНИЕ

Целью данной работы является изучение встроенных средств языка Си для работы с текстом и разработка пользовательского приложения с использованием этих средств.

Для достижения поставленной цели требуется решить следующие задачи:

- изучение теоретических материалов (структуры, функции стандартной библиотеки, цветной вывод текста, широкие символы);
- разработка программного кода в соответствии с тех. заданием;
- сборка программы с использованием заголовочных файлов и мейкфайла;
- тестирование приложения и отладка;
- создание пользовательской инструкции.

1. РАЗРАБОТКА ПРОГРАММНОГО КОДА

1.1. Техническое задание

Вариант 9

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) Изменить все слова в тексте заканчивающиеся на "ться" так чтобы они заканчивались на "тся" и наоборот.
- 2) Вывести все предложения в которых встречается второе слово первого предложения. Данное слово необходимо выделить зеленым цветом.
- 3) Отсортировать предложения по возрастанию количества слов в предложении.
- 4) Удалить все предложения в которых больше 10 слов.

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.

1.2. Разбиение по задачам

Таким образом, код разбивается на следующие задачи:

- ввести текст и организовать его удобное расположение в памяти;
- сравнить предложения между собой без учёта регистра и удалить повторяющиеся, оставив только первое из них;
- реализовать «бесконечный» интерактив с запросом кода действия;
- написать функцию поиска слов по условию и перезаписи их окончаний;
- написать функцию поиска слов по условию и печати результатов поиска с изменением цвета текста;
- написать функцию-компаратор для встроенной функции `qsort()`;
- написать функцию удаления предложений по условию;
- локализовать функционал для работы с широкими символами;
- в конце работы программы очистить выделенную динамическую память;
- также полезно будет иметь функцию печати сохранённого текста, чтобы время от времени давать пользователю возможность видеть, с чем он работает.

Исходя из характера задач, целесообразно разделить их по группам:

- функции ввода-вывода;
- функции обработки текста;
- действия, реализуемые непосредственно в функции `main()`.

1.3. Запись и хранение текста

Размер текста до компиляции не определён и может быть очень большим. Используем динамическую память.

Текст должен быть структурирован удобно для дальнейшей обработки. Разобьём его на предложения, а предложения — на слова.

Реализуем для каждой составной части структурный тип данных: Text, Sentence и Word.

Word — строка широких символов, которую в дальнейшем может потребоваться сравнивать или перезаписывать, поэтому полезно иметь в структуре поля size и len, чтобы не вычислять их заново, а также символ sep — запятая или точка после слова будет храниться именно здесь, чтобы не мешать сравнению слов.

Sentence — массив слов, тоже необходимо сохранить size и len, чтобы знать его границы.

Text — массив предложений и его size и len.

Для удобства (и потому что это не противоречит тех. заданию) было выбрано не разграничивать текст на абзацы — весь текст поступает в программу одной строкой, а символ конца строки служит признаком конца ввода.

Функции readWord(), readSentence() и readText() конструируют структуры из посимвольного ввода, передавая результат работы на уровень выше. Функция printText() печатает текст, чередуя строки-слова и их сохранённые разделители, добавляя после них пробелы, если они не сохранены (поле sep хранит лишь один символ, и если после слова стоит запятая, сохраняется она, а пробел подразумевается и добавляется при печати функцией printText()).

Т. к. в тех. задании точка названа разделителем предложений, а не символом окончания, предусмотрим вариант, когда последнее предложение оканчивается не точкой, и символом конца строки. В этом случае заменим его на точку.

1.4. Общие функции обработки текста

Функция `removeDuplicate()` сравнивает попарно предложения в тексте, начиная с конца. Если для рассматриваемого предложения находится предшествующее ему с тем же количеством слов, то слова в них сравниваются попарно — строка со строкой без учёта регистра (встроенная функция `wscasestr()`), разделитель с разделителем. Если отличия не найдены, рассматриваемое предложение удаляется.

Для удаления предложений реализована отдельная функция `removeSentence()`, поскольку эта операция понадобится в дальнейшем. Функция очищает динамическую память, занимаемую предложением, и сдвигает память в массиве предложений на освободившееся место.

Операция очистки всей динамической памяти, занимаемой текстом, реализуется в функции `freeText()`. Поскольку здесь также требуется очищать память предложений, соответствующую подзадачу было решено вынести в отдельную функцию — `freeSentence()`.

1.5. Функции пользовательской обработки текста

В функции `main()` устанавливается кириллическая локализация (встроенная функция `setlocale()`). Пользовательский интерактив реализован через поясняющие надписи и оператор `switch` в цикле, выполняющемся до ввода кода действия, соответствующего завершению программы. В случае ввода непредусмотренного кода программа уточняет пользователю ожидаемый формат данных.

Первое пользовательское действие выполняет функция `swapWordEnds()`, заменяя окончания слов: «тся» → «ться» и «ться» → «тсся». При необходимости буфер памяти слова расширяется для добавления ещё одного символа. Поиск слова для замены окончания проводится без учёта регистра, а «ь» добавляется того же регистра, что и «с» в окончании.

Функция `findWord()` определяет, существует ли слово, которое необходимо найти, зная порядковый номер предложения и слова в нём. Функция печатает на экран все предложения, где это слово содержится. Сравнение слов происходит без учёта регистра. Найденные слова подсвечиваются, но если после них стоит разделитель, его цвет остаётся по умолчанию. Если слова с таким номером не существует, функция печатает сообщение об этом.

Функция-компаратор `sentCmp()` сравнивает предложения по значению поля `len`, а функция `removeSents()` по значению этого же поля определяет, нужно ли удалять предложение. Сама сортировка реализуется встроенной функцией `qsort()`, вызываемой из `main()`.

2. СБОРКА ПРОГРАММЫ

Функции ввода-вывода `readWord()`, `readSentence()`, `readText()` и `printText()` объявлены в отдельном файле `txtio.c`, которому соответствует заголовочный файл `txtio.h`. Все необходимые для этих функций заголовочные файлы стандартной библиотеки и константные выражения прописаны в `txtio.h`, также в нём описаны структуры для хранения текста.

Функции обработки текста сгруппированы в файле `txtproc.c`: `removeDuplicate()`, `swapWordEnds()`, `findWord()`, `sentCmp()`, `removeSents()`, `removeSentence()`, `freeSentence()`, `freeText()`. Все необходимые заголовочные файлы стандартной библиотеки, прототипы самих функций и описание структур сохранены в `txtproc.h`.

Так как в функцию `main()` будут импортированы оба заголовочных файла, помимо обычной защиты от повторного включения требуется предусмотреть условную компиляцию для описания структур, зависящую от определения одного и того же выражения.

Сборка программы будет осуществляться утилитой `make` в соответствии с инструкциями в мэйкфайле `Makefile`.

Программный код представлен в приложении А настоящего отчёта.

3. ТЕСТИРОВАНИЕ

Результаты тестирования представлены в таблице 1. В целях удобства учитываются только значимые части ввода и вывода. Полностью детали интерактива продемонстрированы на рисунках 1-4 в аннотации.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
Удаление дубликатов предложений. Ввод по разделителям, вывод через пробелы.		
1.	Я,сам.Супе\$5j. сУпЕ\$5J.я, СаМ. супе\$5J	Повторяющиеся предложения удалены: 3 Я, сам. Супе\$5j.
Действие 1: замена окончаний.		
2.	Хочу признаться, КАЖЕТСЯ ДА. МОЖЕТ СТАТСЯ. Отсядь подальше. 1	1. Результат замены на экране: Хочу признаться, КАЖЕТСЯ ДА. МОЖЕТ СТАТЬСЯ. Отсядь подальше.
Действие 2: поиск второго слова первого предложения.		
3.	Для тебя живу. Тебя, тебя люблю, ТЕБЯ. Наверно, тебячата. 2	2. Результат поиска на экране: Для тебя живу. Тебя, тебя люблю, ТЕБЯ. Найдено: 4
4.	Тебя. Тебя лишь я любил. 2	2. Результат поиска на экране: Нечего искать. :(
Действие 3: сортировка по количеству слов.		
5.	А б. АБ. 1, 2, 3. В. Гром. Б. А. 3	3. Результат сортировки на экране: АБ. В. Гром. Б. А. А б. 1, 2, 3.
Действие 4: удаление предложений длиннее 10 слов.		
6.	1 2 3 4 5 6 7 8 9 0. 12 34 56 78 901. 1 2 3 4 5 6 7 8 9 0 1. 4	4. Результат на экране. Удалено предложений: 1 1 2 3 4 5 6 7 8 9 0. 12 34 56 78 901.
Произвольный код действия.		
7.	6	Некорректный номер действия, введите число от 1 до 5.
8.	ш	
Действие 5: выход из программы.		
9.	А, б, в. Агуша. 5	5. Обработка завершена. Результат на экране: А, б, в. Агуша.

4. ПОЛЬЗОВАТЕЛЬСКАЯ ИНСТРУКЦИЯ

«Программа позволяет производить заданные действия над введённым текстом. Какое именно действие из набора применить, определяете Вы.

Шаг первый — запуск приложения

Запустите терминал из папки с файлами приложения. Введите команду «make all && ./Krupin_Nikita_cw».

Шаг второй — ввод текста

Введите в терминал текст и нажмите клавишу «Enter».

Текст вводится в одну строку как последовательность предложений, разделённых точкой или точкой с пробелом.

Слова в предложении разделяются запятой, пробелом или запятой и пробелом.

Шаг третий — выберите действие

Введите номер действия из представленных на экране, нажмите клавишу «Enter».

Результат отобразится на экране.

Шаг четвёртый — продолжаем работу

Вы можете продолжать применять действия друг за другом к введённому тексту. Если в результате действия текст изменился, следующее действие будет применено к новой версии текста.

Шаг пятый — выход из приложения

Чтобы завершить работу, введите номер действия «5».

Вы всегда можете запустить приложение снова и ввести новый текст».

ЗАКЛЮЧЕНИЕ

Изучены новые средства работы с текстом на языке Си. Разработано интерактивное консольное приложение по техническому заданию. Программа собрана, протестирована и отлажена. Написана инструкция для пользователя.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Правила оформления пояснительной записки. // se.moevm.info URL: <http://se.moevm.info/doku.php/courses:programming:report> (дата обращения: 11.12.2021).
2. Функции стандартных библиотек. // wikipedia.org URL: <https://ru.wikipedia.org/wiki/Wchar.h> (дата обращения: 10.12.2021).
3. Функции стандартных библиотек и другие инструкции. // cplusplus.com URL: <https://www.cplusplus.com/reference> (дата обращения: 10.12.2021).

ПРИЛОЖЕНИЕ А

ПРОГРАММНЫЙ КОД

Название файла «main.c»

```
#include <locale.h>
#include "txtio.h"
#include "txtproc.h"

#define BUFF_SIZE 256

int main()
{
    Text text; wchar_t ws[BUFF_SIZE] = L"";
    setlocale(LC_ALL, "");
    fputws(L"\033[0;33mРежим ввода. Введите текст одной
строкой.\n", stdout);
    fputws(L"Разделитель предложений - точка.\n",
stdout);
    fputws(L"Разделитель слов - пробел или запятая.\n
\033[0m\n", stdout);
    text = readText();
    fputws(L"\033[0;32mПовторяющиеся предложения удалены:
", stdout);
    wprintf(L"%d\033[0m\n", removeDuplicate(&text));
    printText(text);
    while (ws[0] != L'5'){
        fputws(L"\033[0;33mРежим обработки текста.
Выберите действие:\n", stdout);
        fputws(L"1 - поменять
окончания \"тс\"-\"тьс\";\n", stdout);
        fputws(L"2 - найти второе слово первого
предложения;\n", stdout);
        fputws(L"3 - сортировать предложения по
возрастанию количества слов;\n", stdout);
        fputws(L"4 - удалить предложения, в которых
больше 10 слов;\n", stdout);
        fputws(L"5 - завершить обработку.\033[0m\n",
stdout);
        fgetws(ws, BUFF_SIZE, stdin);
        switch (ws[0]){
            case L'1':
                fputws(L"\033[0;32m1. Результат замены
на экране:\033[0m\n", stdout);
                swapWordEnds(text);
```



```

        printText(text);
        break;
    case L'2':
        fputws(L"\033[0;32m2. Результат поиска
на экране:\033[0m\n", stdout);
        findWord(0, 1, text);
        break;
    case L'3':
        fputws(L"\033[0;32m3. Результат
сортировки на экране:\033[0m\n", stdout);
        qsort(text.sents, text.len,
sizeof(Sentence), sentCmp);
        printText(text);
        break;
    case L'4':
        fputws(L"\033[0;32m4. Результат на
экране. Удалено предложений: ", stdout);
        wprintf(L"%d\033[0m\n",
removeSents(&text));
        printText(text);
        break;
    case L'5':
        fputws(L"\033[0;32m5. Обработка
завершена. Результат на экране:\033[0m\n", stdout);
        printText(text);
        break;
    default:
        fputws(L"\033[0;31mНекорректный номер
действия, введите число от 1 до 5.\033[0m\n", stdout);
    }
}
freeText(&text);
return 0;
}

```

Название файла «txtio.h»

```
#pragma once

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

#define MEM_STEP 8
#define SENT_SEP L"\\n"
#define WORD_SEP L" ,"

#ifndef __TXT_STRUCT__
#define __TXT_STRUCT__
typedef struct Word{
    wchar_t* wchars;
    wchar_t sep;
    int size, len;
} Word;
typedef struct Sentence{
    Word* words;
    int size, len;
} Sentence;
typedef struct Text{
    Sentence* sents;
    int size, len;
} Text;
#endif

Word readWord();
Sentence readSentence();
Text readText();
void printText(Text);
```

Название файла «txtio.c»

```
#include "txtio.h"

Word readWord(){
    int size = MEM_STEP, len = 0;
    wchar_t* wchars = malloc(size*sizeof(wchar_t));
    wchar_t sep, wc = L' ';

    while (wc == L' ') wc = fgetwc(stdin);
    while (!wcschr(WORD_SEP, wc) && !wcschr(SENT_SEP,
wc)){
        wchars[len++] = wc;
        if (len == size){
            size += MEM_STEP;
            wchars = realloc(wchars,
size*sizeof(wchar_t));
        }
        wc = fgetwc(stdin);
    }
    wchars[len] = L'\0';
    sep = wc;
    return (Word){wchars, sep, size, len};
}

Sentence readSentence(){
    int size = MEM_STEP, len = 0;
    Word* words = malloc(size*sizeof(Word));
    Word ww;

    do{
        ww = readWord();
        if (len == size){
            size += MEM_STEP;
            words = realloc(words, size*sizeof(Word));
        }
        words[len++] = ww;
    } while (!wcschr(SENT_SEP, ww.sep));
    return (Sentence){words, size, len};
}

Text readText(){
    int size = MEM_STEP, len = 0;
    Sentence* sents = malloc(size*sizeof(Sentence));
    Sentence ws;
```

```

do{
    ws = readSentence();
    if (ws.len == 1 && !ws.words[0].len &&
ws.words[0].sep == L'\n'){
        free(ws.words[0].wchars);
        free(ws.words);
        break;
    }
    if(len == size){
        size += MEM_STEP;
        sents = realloc(sents,
size*sizeof(Sentence));
    }
    sents[len++] = ws;
} while (ws.words[ws.len-1].sep != L'\n');
if (ws.words[ws.len-1].sep == L'\n')
    ws.words[ws.len-1].sep = L'.';
return (Text){sents, size, len};
}

void printText(Text text){
    for (Sentence* ws = text.sents; ws < text.sents +
text.len; ws++){
        for (Word* ww = ws->words; ww < ws->words + ws-
>len; ww++){
            fputws(ww->wchars, stdout);
            fputc(ww->sep, stdout);
            if (ww->sep != L' ') fputc(L' ', stdout);
        }
        fputc(L'\n', stdout);
    }
}

```

Название файла «txtproc.h»

```
#pragma once

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

#ifndef __TXT_STRUCT__
#define __TXT_STRUCT__
typedef struct Word{
    wchar_t* wchars;
    wchar_t sep;
    int size, len;
} Word;
typedef struct Sentence{
    Word* words;
    int size, len;
} Sentence;
typedef struct Text{
    Sentence* sents;
    int size, len;
} Text;
#endif

int removeDuplicate(Text*);
void swapWordEnds(Text);
void findWord(int, int, Text);
int sentCmp(const void*, const void*);
int removeSents(Text*);
void removeSentence(int, Text*);
void freeSentence(Sentence);
void freeText(Text*);
```

Название файла «txtproc.c»

```
#include "txtproc.h"

int removeDuplicate(Text* text){
    int cond1, cond2, count = 0;
    Word *w1, *w2;

    for (int ir = text->len-1; ir > 0; ir--){
        for (int il = ir-1; il >= 0; il--){
            if (text->sents[il].len != text->sents[ir].len)
                continue;
            w1 = text->sents[il].words;
            w2 = text->sents[ir].words;
            cond1 = cond2 = 0;
            for (int i = 0; i < text->sents[il].len; i++){
                cond1 = wcscasecmp(w1[i].wchars,
w2[i].wchars);
                cond2 = w1[i].sep != w2[i].sep;
                if (cond1 || cond2) break;
            }
            if (!cond1 && !cond2){
                removeSentence(ir, text);
                count++;
                break;
            }
        }
        return count;
    }
}

void swapWordEnds(Text text){
    wchar_t* wpointer;

    for (Sentence* ws = text.sents; ws <
text.sents+text.len; ws++){
        for (Word* ww = ws->words; ww < ws->words+ws->len; ww++){
            wpointer = ww->wchars + ww->len;
            if (!wcscasecmp(wpointer-3, L"тця")){
                if (ww->size < ww->len+2){
                    ww->size++;
                    ww->wchars = realloc(ww->wchars, ww->size*sizeof(wchar_t));
                }
            }
        }
    }
}
```

```

        wpointer = ww->wchars + ww->len;
    }
    memmove(wpointer-1, wpointer-2,
3*sizeof(wchar_t));
    if (iswupper(wpointer[-1]))
        wpointer[-2] = L'Ь';
    else wpointer[-2] = L'ь';
    ww->len++;
} else if (!wcscasecmp(wpointer-4, L"ться"))
{
    memmove(wpointer-3, wpointer-2,
3*sizeof(wchar_t));
    ww->len--;
}
}
}

```

```

void findWord(int is, int iw, Text text){
    int print, count = 0;
    wchar_t* fw;

    if (is<text.len && iw<text.sents[is].len){
        fw = text.sents[is].words[iw].wchars;
        for (Sentence* ws = text.sents; ws <
text.sents+text.len; ws++){
            print = 0;
            for (Word* ww = ws->words; ww < ws-
>words+ws->len; ww++){
                if (!wcscasecmp(ww->wchars, fw)){
                    print = 1;
                    break;
                }
            }
            if (!print) continue;
            for (Word* ww = ws->words; ww < ws-
>words+ws->len; ww++){
                if (!wcscasecmp(ww->wchars, fw)){
                    count++;
                    fputws(L"\033[0;32m", stdout);
                }
                fputws(ww->wchars, stdout);
                fputws(L"\033[0m", stdout);
                fputwc(ww->sep, stdout);
                if (ww->sep != L' ') fputwc(L' ',
stdout);
            }
        }
    }
}

```

```

        fputwc(L'\n', stdout);
    }
    fputws(L"\033[0;32mНайдено: ", stdout);
    wprintf(L"%d\033[0m\n", count);
} else fputws(L"\033[0;31mНечего искать. :(\033[0m\n",
n", stdout);
}

int sentCmp (const void* arg1, const void* arg2){
    Sentence* ws1 = (Sentence*)arg1;
    Sentence* ws2 = (Sentence*)arg2;

    return ws1->len - ws2->len;
}

int removeSents(Text* text){
    int count = 0;

    for (int i = text->len-1; i >= 0; i--){
        if (text->sents[i].len > 10){
            removeSentence(i, text);
            count++;
        }
    }
    return count;
}

void removeSentence(int i, Text* text){
    freeSentence(text->sents[i]);
    memmove(text->sents+i, text->sents+i+1, (text->len-i-1)*sizeof(Sentence));
    text->len--;
}

void freeSentence(Sentence ws){
    for (Word* ww = ws.words; ww < ws.words + ws.len; ww++)
        free(ww->wchars);
    free(ws.words);
}

void freeText(Text* text){
    for (int i = 0; i < text->len; i++)
        freeSentence(text->sents[i]);
    free(text->sents);
    text->sents = NULL;
}

```



```
    text->size = 0;  
    text->len = 0;  
}
```

Название файла «Makefile»

```
all: Krupin_Nikita_cw
```

```
Krupin_Nikita_cw: main.o txtio.o txtproc.o  
    gcc main.o txtio.o txtproc.o -o Krupin_Nikita_cw
```

```
main.o: main.c txtio.h txtproc.h  
    gcc -c main.c
```

```
txtio.o: txtio.c txtio.h  
    gcc -c txtio.c
```

```
txtproc.o: txtproc.c txtproc.h  
    gcc -c txtproc.c
```

```
clean:  
    rm -rf *.o Krupin_Nikita_cw
```