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.

Студентка гр. 0382 _____ Кривенцова. Л.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студентка Кривенцова Л.С.

Группа 0382

Тема работы: Обработка строк на языке Си.

Исходные данные:
Программа должна быть написана на языке Си, разделена на функции, выполняющие подзадачи, осуществляющие обработку подаваемого на вход программы текста, который должен быть сохранён в виде динамического массива. Ввод, обработка и вывод данных должны осуществляться с помощью функций стандартной библиотеки, а также с использованием структур.
Содержание пояснительной записки:
разделы «Аннотация», «Содержание», «Введение» («Цель и задачи»), «Выполнение работы». «Примеры работы программы», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 02.11.2020

Дата сдачи реферата: 14.12.2020

Дата защиты реферата: 26.12.2020

Студентка _____ Кривенцова. Л.С.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

Курсовая работа представляет собой реализацию программы на языке Си в качестве решения задачи по обработке текста. Для работы с текстом произвольной длины использовались функции стандартных библиотек, динамическая память и структуры.

После получения входных данных (текста) программа печатает подсказку, предоставляя выбор одного из следующих доступных действий. Работа программы также предусматривает ошибочные действия со стороны пользователя (такие, как некорректный ввод команды), уведомляя его посредством выводимых в терминал сообщений. Далее выполняются требуемая обработка текста и вывод преобразованных данных.

Исходный код работы программы приведён в приложении А.

Пример работы программы приведён в приложении Б.

SUMMARY

The course work is an implementation of a program in C as a solution to the problem of text processing. Standard library functions, dynamic memory, and structures were used to work with text of free length.

After receiving the input data (text), the program prints a hint, providing a choice of one of the following available actions. The program also provides for incorrect actions on the part of the user (such as incorrect command input), notifying him by showing a messages on terminal. Then the required text processing and output of the converted data are performed.

The source code of the program is given in Appendix A.

An example of how the program works is given in Appendix B.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является написание программы, исполняющей одно из доступных действий по редактированию текста (по выбору пользователя). Реализованный код должен работать с применением динамической памяти и структур, для хранения и обработки необходимых данных.

Для достижения поставленной цели требовалось решить следующие задачи: изучение теоретического материала, разработка кода, разбиение на функции, группировка функций в несколько файлов, сборка Makefile, тестирование.

Итог выполнения работы представляет собой корректно работающую программу на языке Си, которая выполняет считывание исходных данных с клавиатуры (потока stdin), печатает подсказку, предлагающую пользователю выбрать одно из возможных действий, которое далее производится над введённым текстом (выбором пользователя также может быть выход из программы). Для поддерживания кириллического букв используются широкие символы типа данных wchar_t и функция setlocale(). После обработки динамического массива, в котором хранится и преобразовывается текст, он выводится в терминал (поток stdout). Завершают программу действия по очистке выделенной ранее динамической памяти (посредством функции free).

Для отладки созданной программы её тестирование осуществлялось на интегрированной среде разработки для языков программирования Си Clion.

1. ЗАДАНИЕ

Вариант 8.

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. Для каждого предложения вывести строку-дату вида “ДД-ММ-ГГГГ”, где день - количество слов в предложении, месяц - наибольшая длина слова в предложении, год - общее количество символов в предложении + 1900.
2. Вывести предложения так, чтобы слова шли в обратном порядке.
3. Отсортировать предложения по длине первого слова в предложении.
4. Удалить все предложения у которых все слова имеют длину не больше 3 символов.

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. ВЫПОЛНЕНИЕ РАБОТЫ

2.1. Изучение основных теоретических положений и требований к выполнению работы.

2.2. Разработка кода.

2.2.1. Ход решения:

Используется стандартная библиотека Си, её заголовочные файлы *stdio.h, stdlib.h; wchar.h, locale.h, wctype.h* — для работы с широкими символами (в данном случае — кириллическими буквами).

Объявляются две структуры — *Text* и *Sentence*, для хранения текста и предложений (и информации о них) соответственно. Вызывается функция *setlocale(LC_ALL, "")* , позволяющая задать локализацию программы, в том числе установить локальный языковой стандарт — русский.

В стандартный поток вывода *stdout* печатается приветственное сообщение с первичной инструкцией для пользователя. Создаётся переменная типа *Text*, для поля (динамического массива структур *Sentence*) которой выделяется память с помощью *malloc.* Запускается цикл *do...while* считывания предложений. В нём создаётся переменная — объект структуры *Sentence*, инициализируются её поля, для которых также динамически выделяется память посредством *malloc.*

На вход программы подаётся текст. Предложения считываются посимвольно в поле объекта структуры *Sentence* — *sent.phrase* (динамический массив указателей на *wchar_t*) из стандартного потока ввода с помощью функции *fgetwc(stdin)*. Все незначащие пробелы (стоящие перед первым словом предложения) не записываются. Ввод предложения оканчивается символом *'.'*, а ввод текста — символом переноса строки *'\n'*, проверка осуществляется посимвольно в циклах *while*. Сразу после считывания одного предложения оно проверяется на повтор (без учёта регистра), и если оказывается, что такое предложение ещё не встречалось в тексте — экземпляр структуры *Sentence* (- *sent*), в полях которой содержится предложение и некоторая информация о нём — записывается в ячейку динамического массива — поля объекта структуры *Text* —

tex.text, в ином случае цикл считывания предложений продолжается, не сохраняя текущее.

После считывания каждого символа выполняется проверка на остаток выделенной динамической памяти для хранения предложения и в случае нехватки производится довыделение с помощью *realloc*. После сохранения предложения выполняется та же проверка, но для хранения текста (массива структур *Sentence*).

После того, как был встречен символ окончания ввода, начинается цикл общения с пользователем: печатается подсказка, предлагающая задать символ, выбирая одну из предложенных команд по обработке текста. Символ считывается через функцию *scanf*. Цикл завершается только в том случае, когда пользователь введёт символ, являющийся командой к окончанию программы (*'\0'*). При помощи оператора *switch*, в котором описаны 6 случаев (5 из которых — команда, заданная символом, и *default*, в случае некорректного ввода), в зависимости от введённого символа выполняется команда по обработке текста (вызывается одна из функций *str_data, first_word, right_proposal, reverse*), либо команды вывода текста и выхода из программы.

Память, выделенная динамически в программе ранее, освобождается при выполнении алгоритма корректного выхода из программы (после того, как пользователь ввёл (*'\0'*).

Входные данные:

Программа корректно работает со следующими входными данными: подаётся текст (предложения, разделённые точкой; все пробелы, стоящие перед началом предложения — не учитываются при обработке текста). 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. Окончанием текста является символ переноса строки - *'\n'*. Знак окончания текста не может быть встречен внутри предложения.

2.2.2. Структуры.

Структура *Sentence* — структура для хранения предложения и некоторой информации о нём. Имеет 6 полей:

words (тип *wchar_t* **) - двумерный динамический массив для работы со словами предложения;

phrase (тип *wchar_t* *) - динамический массив для хранения текста предложения;

lent (тип *int*) — целое число, хранящее общее количество символов в предложении.

countwords (тип *int*) — целое число, хранящее количество слов в предложении;

maxword (тип *int*) — целое число, хранящее наибольшую длину слова в предложении;

right - (тип *int*) — целое число, выполняющее роль логической переменной и хранящее состояние «1» или «0», в зависимости от того, удовлетворяет предложение требованиям или его нужно удалить.

Структура *Text* — структура для хранения текста и некоторой информации о нём. Имеет 2 поля:

text (тип *struct Sentence**) - динамический массив предложений (структур *Sentence*);

len (тип *int*) — целое число, хранящее общее количество предложений в тексте.

2.2.3. Переменные.

Главной функции *main()*:

wchar_t command; - переменная, в которую считывается номер команды, которую следует выполнить.

wchar_t sym; - переменная, в которую считывается следующий символ предложения.

int i; - целочисленная переменная, использующаяся для итераций цикла *for*.

int k; - целочисленная переменная, использующаяся для итераций цикла *for*.

int addmemory; - целочисленная переменная, использующаяся для довыделения динамической памяти.

int flag; - целочисленная переменная, выполняющая роль логической переменной и хранящая состояние «1» или «0», в зависимости от того, должен завершиться ввод текста или нет.

int same; - целочисленная переменная, выполняющая роль логической переменной и хранящая состояние «1» или «0», в зависимости от того,является предложение повтором одного из предыдущих или нет.

int yes; - целочисленная переменная, выполняющая роль логической переменной и хранящая состояние «1» или «0», в зависимости от того, нужно включать предложение в текст или нет.

struct Sentence sent; - переменная, являющаяся объектом структуры *Sentence*, хранящая текущее информацию о текущем предложении.

struct Text tex; - переменная, являющаяся объектом структуры *Text*, хранящая динамический массив структур предложений и количество предложений на текущий момент.

struct Sentence ex; - переменная, являющаяся объектом структуры *Sentence*, использующаяся для хранения вспомогательного предложения (для вывода в цикле, для сравнения текущего предложения с предыдущими на предмет повтора).

Функции *str_data(struct Sentence sent):*

int day; - целочисленная переменная, хранящая количество слов в предложении - «день» для формата строки-даты.

int month; - целочисленная переменная, хранящая наибольшая длина слова в предложении - «месяц» для формата строки-даты.

int year; - целочисленная переменная, хранящая значение (1900 + общее количество символов в предложении) - «год» для формата строки-даты.

Функции *first_word(const void * item1, const void * item2):*

int lenitem1; - целочисленная переменная, хранящая длину первого слова одного из сравниваемых предложений предложения.

int lenitem2; - целочисленная переменная, хранящая длину первого слова второго из сравниваемых предложений предложения.

int i; - целочисленная переменная, использующаяся для итераций цикла *for*.

struct Sentence phrase1;* - указатель на структуру, содержащую одно из сравниваемых в функции предложений.

struct Sentence phrase2;* - указатель на структуру, содержащую второе из сравниваемых в функции предложений.

Функции *right_proposal(struct Sentence sent):*

int count; - целочисленная переменная, хранящая длину слова, последнего найденного в текущей итерации цикла проходящего по символам предложения;

int i; - целочисленная переменная, использующаяся для итераций цикла *for*.

Функции *reverse(struct Sentence sent):*

int quit; - целочисленная переменная, хранящая длину текущего слова предложения;

*wchar_t *new;* - динамический массив широких символов; строка, в которую сохраняется отредактированное предложение (в которое слова записываются в обратном порядке);

int count; - целочисленная переменная, хранящая индекс строки *new*, в который следует записать следующий символ при копировании строк;

int i; - целочисленная переменная, использующаяся для итераций цикла *for*;

int k; - целочисленная переменная, использующаяся для итераций цикла *for*;

*wchar_t *pt;* - указатель на объект типа *wchar_t* *, который используется *wcstok* для хранения своего внутреннего состояния.

2.2.4. Функции.

int main().

В самом начале функция печатает на экран первичное обращение к пользователю. После выделения памяти для динамических массивов, функция реализует ввод текста (по предложениям) из стандартного потока ввода в цикле *do..while(),* и соответственно, ввод предложений посимвольно во вложенном цикле *do..while()*. Помимо сохранения предложений, в циклах осуществляется проверка на то, хватает ли выделенной динамической памяти и в противном случае память перевыделяется с помощью *realloc*. После считывания каждого предложения также запускается цикл *for*, в котором предложение проверяется на дублирование одного из введённых ранее. Если оно является повтором, то структура, хранящая его (*sent*), не записывается в динамический массив структур *tex.text.*

После получения знака завершения текста — символа переноса строки, запускается цикл общения с пользователем. Функция осуществляет ввод команды с клавиатуры, в зависимости от которой вызывает нужную функцию через оператор выбора *switch* и описанные в нём случаи *case* и *default*. Освобождает выделенную ранее динамическую память и завершает программу при вводе команды «0».

Возвращает 0 при корректной работе. Не имеет аргументов.

void str_data(struct Sentence sent).

Функция принимает объект структуры *Sentence* в качестве аргумента, но ничего не возвращает. Она высчитывает три параметра — день, месяц и год, и с помощью форматного вывода функции стандартной библиотеки *wprintf* выводит строку вида ДД-ММ-ГГГГ. Перед выводом реализованы проверки на добавление в строку значащих нулей.

int first_word(const void * item1, const void * item2).

Функция реализована для подачи её в качестве аргумента в функцию *qsort(),* вызываемой в *case* L'3' оператора *switch* в функции *main()*.

Функция принимает на вход два аргумента, имеющих тип *const void* * (чтобы удовлетворять требованиям *qsort*). Для работы с переданными в функцию объектами, необходимо выполнить их явное преобразование к типу (*struct Sentence**). Далее функция считает длины первых слов двух сравниваемых предложений (в переменные *int lenitem1* и *int lenitem2*), обращаясь к полям *phrase* объектов (с помощью - >). Функция возвращает разность длин первых слов *lenitem1* и *lenitem2*: возвращает значение «0», если длины равны и функция *qsort* не должна менять порядок переданных двух объектов; значение <0 , если переданные в функцию объекты также удовлетворяют требованиям сортировки и их не нужно менять местами; значение >0, если переданные в функцию объекты не удовлетворяют требованиям сортировки и функция *qsort* должна поменять их местами.

struct Sentence right_proposal(struct Sentence sent).

Функция принимает на вход аргумент - объект структуры *Sentence* (*sent*), и возвращает его же изменённую версию. Функция изменяет значение поля объекта *sent* — *sent.right*, в зависимости от того, удовлетворяет предложение требованию, заданному в условии подзадачи (хотя бы одно слово в предложении должно иметь длину больше трёх символов) или нет. Для этого в цикле for осуществляется поиск самого длинного слова в предложении (его длина хранится в поле объекта *sent* - *sent.maxword*), затем с помощью оператора *if* проверяется условие.

struct Sentence reverse(struct Sentence sent).

Функция принимает объект *struct Sentence* — *sent* в качестве аргумента и возвращает его отредактированную версию (в которой слова в предложении стоят в обратном порядке).

С помощью *malloc* выделяется память для строки *wchar_t * - new*. Затем заполняется двумерный динамический массив слов предложения — *sent.words* (реализуется в цикле *while*, функцией *wcstok*). Далее в цикле *for* (со вложенным циклом *for*) формируется строка *new* — которая заменит нынешнюю строку, хранящуюся в поле *sent.phrase*. В неё осуществляется посимвольное копирова-

ние слов предложения - *sent.words*, между которыми вставляется разделитель ' '. Полю *sent.phrase* присваивается сформированная строка *new* и функция возвращает изменённый объект структуры *Sentence — sent*.

2.2.5. Линковка.

Была произведено распределение функций программы в несколько файлов.

Заголовочные файлы (одноименных функций):

- first_word.h;
- right_proposal.h;
- reverse.h;
- str_data.h;

Файлы реализации(одноименных функций):

- main.c;
- first_word.c;
- right_proposal.c;
- reverse.c;
- str_data.c;

Сборка, включающая создание необходимых объектных файлов (main.o, first_word.o, right_proposal.o, reverse.o, str_data.o) и исполняемого файла main осуществляется с помощью Makefile (см. код Makefile в Приложении А.)

2.3. Тестирование.

Тестирование программы происходило в интегрированной среде разработки для языков программирования Си Clion.

Результаты тестирования представлены в табл.1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Dieu,quelle virulente sortie. Князь Василий говорил всегда лениво,как актер говорит роль старой пиесы. Анна Павловна Шерер,напротив,несмотря на свои сорок лет,была преисполне-	Я ничего не понимаю,может быть,но Австрия никогда не хотела и не хочет войны. Она о нас. Наш благодетель знает свое высокое призвание и будет верен ему. Вот	Программа считала текст и команды, выполнила сортировку по длине первого слова

	на оживления и порывов. Я ничего не понимаю,может быть,но Австрия никогда не хотела и не хочет войны. Она о нас. Россия одна должна быть спасительницей Европы. Наш благодетель знает свое высокое призвание и будет верен ему. Вот одно во что я верю. 3 5 0	одно во что я верю. Dieu,quelle virulente sortie. Анна Павловна Шерер,напротив,несмотря на свои сорок лет,была преисполнена оживления и порывов. Князь Василий говорил всегда лениво,как актер говорит роль старой пиесы. Россия одна должна быть спасительницей Европы.	предложения, вывела преобразованный текст на экран и корректно завершилась.
2.	Dieu,quelle virulente sortie. Князь Василий говорил всегда лениво,как актер говорит роль старой пиесы. Анна Павловна Шерер,напротив,несмотря на свои сорок лет,была преисполнена оживления и порывов. Я ничего не понимаю,может быть,но Австрия никогда не хотела и не хочет войны. Она о нас. Россия одна должна быть спасительницей Европы. Наш благодетель знает свое высокое призвание и будет верен ему. Вот одно во что я верю. 4 2 5 0	sortie virulente quelle Dieu. пиесы старой роль говорит актер как лениво всегда говорил Василий Князь. порывов и оживления преисполнена была лет сорок свои на несмотря напротив Шерер Павловна Анна. войны хочет не и хотела не никогда Австрия но быть может понимаю не ничего Я. Европы спасительницей быть должна одна Россия. ему верен будет и призвание высокое свое знает благодетель Наш. верю я что во одно Вот.	Программа считала текст и команды, удалила все предложения (их встретилось 1), у которых все слова имеют длину не больше трёх символов, переставила слова в предложении в обратном порядке, вывела преобразованный текст на экран и корректно завершилась.
3.	Молодая княгиня Болконская приехала с работой в шитом золотом бархатном мешке. Как это бывает у вполне привлекательных женщин,недостаток ее — короткость губы и полуоткрытый рот — казались ее особенною,собственно ее красотой. Всем было весело смотреть на эту полную здоровья и живости хорошенькую будущую мать,так легко переносившую свое положение. F 1 0	Данные некорректны Молодая княгиня Болконская приехала с работой в шитом золотом бархатном мешке. 11-10-1977 Как это бывает у вполне привлекательных женщин,недостаток ее — короткость губы и полуоткрытый рот — казались ее особенною,собственно ее красотой. 22-15-2044 Всем было весело смотреть на эту полную здоровья и живости хорошенькую будущую мать,так легко переносившую свое положение. 18-12-2021	Программа считала текст, получила на вход некорректную команду, о чём уведомила пользователя; считала новые введённые команды, вывела строку-дату для каждого предложения и корректно завершилась.

ЗАКЛЮЧЕНИЕ

Были изучены основы языка программирования Си, применены в ходе работы его составляющие, такие как функции стандартных библиотек, методы работы с динамической памятью и структуры. Реализовано решение задания к курсовой работе, результатом чего является программа, осуществляющая считывание и проверку введённых данных, обработку и вывод текста согласно указанным условиям. Код разделён на подзадачи (отдельные функции, объединённые в файлы), его сборка осуществляется с помощью написанного Makefile. Обмен информацией с пользователем происходит через терминал, посредством стандартных потоков ввода и вывода. Включенная в программу проверка пресекает введение некорректных данных, завершая уведомляя об этом пользователя.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Б.В. Керниган, Д.М. Ричи. Язык программирования Си.: Санкт-Петербург: Невский диалект, 2000. 352 с.

2. Сайт C++ Resources Network URL: http://cplusplus.com (дата обращения: 04.12.2020).

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <wchar.h>
#include <locale.h>
#include <wctype.h>
#include "first_word.h"
#include "right_proposal.h"
#include "reverse.h"
#include "str_data.h"
#include "structs.h"

int main() {
    int flag = 1, yes = 1, same = 0;
    wchar_t sym, command;
    int addmemory = 100;
    struct Sentence ex;
    setlocale(LC_ALL, "");
    struct Text tex;
    wprintf(L"Вас приветствует программа по обработке текста!\n"
         "Пожалуйста, введите текст, который следует отредактировать:\n"
         "(Чтобы закончить ввод, следует нажать ""enter"", спасибо).\n");

    tex.text = malloc(addmemory * sizeof(struct Sentence));
    tex.len = 0;
    do {
        struct Sentence sent;
        sent.phrase = malloc(addmemory * sizeof(wchar_t));
        sent.words = malloc(addmemory * sizeof(wchar_t *));
        addmemory *= 2;
        sent.lent = 0;
        sent.maxword = 0;
        sent.countwords = 0;
        sent.right = 1;
        sym = (wchar_t) fgetwc(stdin);
        if (sym == L'\n') flag = 0;
        else {
            while (sym == L' ') sym = (wchar_t) fgetwc(stdin);
            sent.phrase[sent.lent++] = sym;
            do {
                sym = (wchar_t) fgetwc(stdin);
                sent.phrase[sent.lent++] = sym;
                if (sent.lent == addmemory) {
                    addmemory *= 2;
                    sent.phrase = realloc(sent.phrase, addmemory *
sizeof(wchar_t));
                    sent.words = realloc(sent.words, addmemory *
sizeof(wchar_t *));
                }
            } while (sym != L'.');
```

```
            sent.phrase[sent.lent] = L'\0';
            for (int k = 0; k < tex.len; k++) {
                ex = tex.text[k];
                same = 1;
                for (int j = 0; j < sent.lent && j < ex.lent; j++) {
                    if ((wchar_t) towlower(sent.phrase[j]) != (wchar_t)
towlower(ex.phrase[j])) {
                        same = 0;
                        break;
                    }
                }
                if (same) {
                    yes = 0;
                    break;
                }
            }
            if (yes == 1) {
                tex.text[tex.len++] = sent;
                if (tex.len == addmemory) {
                    addmemory *= 2;
                    tex.text = realloc(tex.text, addmemory);
                }
            }
        }
        yes = 1;
    } while (flag == 1);
    while (command != L'0') {
        wprintf(L"\nЧтобы выбрать одну из возможных команд, "
                "введите в терминал соответствующую цифру:\n"
                "0 - Выйти из программы;\n"
                "1 - Вывести строку-дату для каждого предложения;\n"
                "2 - Отредактировать предложения так, чтобы слова шли в
обратном порядке;\n"
                "3 - Отсортировать предложения по длине первого слова в
предложении;\n"
                "4 - Удалить предложения, у которых все слова имеют длину
не больше трёх символов;\n"
                "5 - Вывести преобразованный текст на экран.\n");
        wscanf(L"%ls", &command);
        switch (command) {
            case L'0':
            {for (int i = 0; i <= tex.len-1; i++){
                    free(tex.text[i].phrase);
                    free(tex.text[i].words);
                }
                free(tex.text);
            }
                break;
            case L'1': {for (int k = 0; k<=tex.len-1; k++){
                    wprintf(tex.text[k].phrase);
                    ex =  right_proposal(tex.text[k]);
                    str_data(ex);
                }
            }
                break;
            case L'2': {
                for (int k = 0; k<=tex.len-1; k++){
                    tex.text[k] = reverse(tex.text[k]);
```

```
                }
            }
                break;
            case L'3': {
                qsort( tex.text, tex.len, sizeof(struct Sentence),
first_word);
            }
                break;
            case L'4': {
                for (int k = 0; k<=tex.len-1; k++){
                    ex =  right_proposal(tex.text[k]);
                    if (ex.right == 0) {
                        for (int r = k; r<tex.len-1; r++) {
                            tex.text[r].phrase = tex.text[r+1].phrase;
                            tex.text[r].lent = tex.text[r+1].lent;
                        }
                        free(tex.text[tex.len].phrase);
                        tex.len -=1;
                    }
                }
            }
                break;
            case L'5': {
                for (int i = 0; i <= tex.len-1; i++)
                    wprintf(L"%ls ", tex.text[i].phrase);
            }
                break;
            default:
                wprintf(L"Данные некорректны");
                break;
        }
    }
    return 0;
}
```

Название файла: first_word.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <wchar.h>
#include <locale.h>
#include <wctype.h>
#include "structs.h"
#include "first_word.h"
int first_word(const void * item1, const void * item2){
    int lenitem1 = 0, lenitem2 = 0,i;
    struct Sentence* phrase1 = (struct Sentence*)item1;
    struct Sentence* phrase2 = (struct Sentence*)item2;
    for (i = 0; phrase1->phrase[i]!=L' ' && phrase1->phrase[i]!=L','; i+
+) lenitem1++;
    for (i = 0; phrase2->phrase[i]!=L' ' && phrase1->phrase[i]!=L','; i+
+) lenitem2++;
    return (lenitem1 - lenitem2);
}
```

Название файла: first_word.h

```
int first_word(const void * item1, const void * item2);
```

Название файла: reverse.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <wchar.h>
#include <locale.h>
#include <wctype.h>
#include "structs.h"
#include "reverse.h"
struct Sentence reverse(struct Sentence sent) {
    wchar_t *pt;
    int i, k, count = 0;
    wchar_t *new = malloc(sent.lent*sizeof(wchar_t *));
    if (sent.lent > 1) {
        sent.phrase[sent.lent - 1] = L'\0';
        sent.phrase = wcstok(sent.phrase, L" ,", &pt);
        while (sent.phrase != NULL) {
            sent.words[sent.countwords++] = sent.phrase;
            sent.phrase = wcstok(NULL, L" ,", &pt);
        }
        for (k = sent.countwords-1; k >= 0; k--){
        int quit = wcslen(sent.words[k]);
           if (count>0) { new[count]=L' ';
            count++;}
        for ( i = 0; i < quit; i++){
            new[count]= sent.words[k][i];
        count++;}
        }
        new[count++]=L'.';
        new[count]=L'\0';
        sent.phrase = new;
        sent.lent =wcslen(sent.phrase);
    }
    return sent;
}
```

Название файла: reverse.h

```
struct Sentence reverse(struct Sentence sent);
```

Название файла: right_proposal.c

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

```
#include "right_proposal.h"
struct Sentence right_proposal(struct Sentence sent){
    int  count = 0;
    for (int i = 0; i < sent.lent; i++){
        if (sent.phrase[i]== L' ' || sent.phrase[i] == L',' ||
sent.phrase[i] == L'.'){
            if (count>sent.maxword) sent.maxword = count;
            count = 0;
            sent.countwords++;
        }
        else  count++;
    }
    if (sent.maxword>3)  sent.right = 1;
    else  sent.right = 0;
    return sent;
}
```

Название файла: right_proposal.h

```
struct Sentence right_proposal(struct Sentence sent);
```

Название файла: str_data.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <wchar.h>
#include <locale.h>
#include <wctype.h>
#include "structs.h"
#include "str_data.h"
void str_data(struct Sentence sent){
    int day, month, year;
    day = sent.countwords;
    month = sent.maxword;
    year = 1900 + sent.lent - 1;
    if (day < 10){
        if(month < 10){ wprintf(L" 0%d-0%d-%d\n", day, month, year);}
        else{wprintf(L" 0%d-%d-%d\n", day, month, year);}
    }
    else if(month < 10){wprintf(L" %d-0%d-%d\n", day, month, year);}
    else{wprintf(L" %d-%d-%d\n", day, month, year);}
}
```

Название файла: str_data.h

```
void str_data(struct Sentence sent);
```

Название файла: structs.h

```
struct Sentence{
    wchar_t **words;
    wchar_t *phrase;
    int lent, right, countwords, maxword;
```

```
};

struct Text{
    struct Sentence* text;
    int len;
};
```

Название файла: Makefile

```
main: main.o first_word.o right_proposal.o reverse.o str_data.o
	gcc main.o first_word.o right_proposal.o reverse.o str_data.o -o
main
main.o: main.c
	gcc -c main.c
first_word.o: first_word.c
	gcc -c first_word.c
right_proposal.o: right_proposal.c
	gcc -c right_proposal.c
reverse.o: reverse.c
	gcc -c reverse.c
str_data.o: str_data.c
	gcc -c str_data.c
clean:
	rm *.o main
```

ПРИЛОЖЕНИЕ Б

ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ.

Пример вывода после вызова make:

```
kaliningrad@kaliningrad-Surface-Laptop:~/Документы$ make
gcc -c main.c
gcc -c first_word.c
gcc -c right_proposal.c
gcc -c reverse.c
gcc -c str_data.c
gcc main.o first_word.o right_proposal.o reverse.o str_data.o -o main
```

Пример вывода терминала во время работы программы:

```
kaliningrad@kaliningrad-Surface-Laptop:~/Документы$ ./main
Вас приветствует программа по обработке текста!
Пожалуйста, введите текст, который следует отредактировать:
(Чтобы закончить ввод, следует нажать enter, спасибо).
Dieu,quelle virulente sortie. Князь Василий говорил всегда лениво,как актер говорит роль старой пиесы. Анна Павловна Шерер,напротив,несмотря на свои сорок лет,была преисполнена оживления и порывов.      Я ничего не понимаю,может быть,но Австрия никогда не хотела и не хочет войны. Она о нас. Россия одна должна быть спасительницей Европы. Наш благодетель знает свое высокое призвание и будет верен ему. Вот одно во что я верю.

Чтобы выбрать одну из возможных команд, введите в терминал соответствующую цифру:
0 - Выйти из программы;
1 - Вывести строку-дату для каждого предложения;
2  - Отредактировать предложения так, чтобы слова шли в обратном порядке;
3 - Отсортировать предложения по длине первого слова в предложении;
4 - Удалить предложения, у которых все слова имеют длину не больше трёх символов;
5 - Вывести преобразованный текст на экран.
2

Чтобы выбрать одну из возможных команд, введите в терминал соответствующую цифру:
0 - Выйти из программы;
1 - Вывести строку-дату для каждого предложения;
2  - Отредактировать предложения так, чтобы слова шли в обратном порядке;
3 - Отсортировать предложения по длине первого слова в предложении;
4 - Удалить предложения, у которых все слова имеют длину не больше трёх символов;
5 - Вывести преобразованный текст на экран.
3

Чтобы выбрать одну из возможных команд, введите в терминал

соответствующую цифру:
0 - Выйти из программы;
1 - Вывести строку-дату для каждого предложения;
```

2 - Отредактировать предложения так, чтобы слова шли в обратном порядке;
3 - Отсортировать предложения по длине первого слова в предложении;
4 - Удалить предложения, у которых все слова имеют длину не больше трёх символов;
5 - Вывести преобразованный текст на экран.
5
нас о Она. ему верен будет и призвание высокое свое знает благодетель Наш. верю я что во одно Вот. пиесы старой роль говорит актер как лениво всегда говорил Василий Князь. войны хочет не и хотела не никогда Австрия но быть может понимаю не ничего Я. sortie virulente quelle Dieu. Европы спасительницей быть должна одна Россия. порывов и оживления преисполнена была лет сорок свои на несмотря напротив Шерер Павловна Анна.
Чтобы выбрать одну из возможных команд, введите в терминал соответствующую цифру:
0 - Выйти из программы;
1 - Вывести строку-дату для каждого предложения;
2 - Отредактировать предложения так, чтобы слова шли в обратном порядке;
3 - Отсортировать предложения по длине первого слова в предложении;
4 - Удалить предложения, у которых все слова имеют длину не больше трёх символов;
5 - Вывести преобразованный текст на экран.
0