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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: обработка строк на языке Си

Студент гр. 0382 _____ Кондратов Ю.А

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Кондратов Ю.А

Группа 0382

Тема работы: обработка строк на языке Си

Исходные данные:

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) “Раскрасить” каждое слово в зависимости от остатка от деления его длины на 4. Если остаток равен 0 - красный цвет, 1 - синий, 2 - зеленый, 3 - желтый.

2) Распечатать каждое слово которое начинается и заканчивается на заглавную букву и номера предложений в которых оно встречается .

3) Отсортировать предложения по длине последнего слова в предложении.

4) Удалить все числа из предложений. Число - набор подряд идущих цифр,

перед и после которого стоят пробелы.

Все сортировки и операции со строками должны осуществляться с использованием функций стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Каждую подзадачу следует вынести в отдельную функцию, функции сгруппировать в несколько файлов (например, функции обработки текста в один, функции ввода/вывода в другой). Также, должен быть написан Makefile.
Содержание пояснительной записки:

Отчет должен содержать подробное описание выполненной вами работы, программной реализации того или иного функционала.

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания:

Дата сдачи работы:

Дата защиты работы:

Студент _____ Кондратов Ю.А.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В процессе выполнения курсовой работы была реализована программа для обработки текст на языке Си. Для хранения текста использовались структуры. Обработка и вывод текста производились при помощи использования функций следующих стандартных заголовочных файлов: stdio.h, stdlib.h, wchar.h, wctype.h, locale.h. Для удобства пользователя реализован вывод на консоль контекстного меню выбора необходимой опции обработки текста. При некорректном вводе номер опции выводится сообщение об ошибке. Также был написан makefile для удобной сборки программы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

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

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

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

1. ЦЕЛЬ И ЗАДАЧИ

Цель: разработка стабильной программы, способной выполнять считывание текста и его обработку в соответствии с заданием.

Для достижения цели необходимо выполнить следующие задачи:

- Реализовать считывание текста;
- Написать функции обработки текста в соответствии с заданием;
- Создать makefile;
- Произвести отладку программы.

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1 Функции и структуры для считывания текста

Для считывания текста реализованы структуры Sentence и Text:

```
struct Sentence {
    wchar_t *sentence;
    wchar_t *sent_copy;
    wchar_t **words;
    wchar_t *first_word;

    int is_final;
    int sent_len;
    int words_count;
};

struct Text {
    struct Sentence *sentences;
    int text_len;
};
```

Элементы структуры Sentence:

- wchar_t *sentence — указатель на начала предложения;
- wchar_t *sent_copy — указатель на предложение-копию (будет использован при разделении предложения на слова);
- wchar_t **words — массив указателей на начала слов в предложении;
- wchar_t *first_word — указатель на начала первого слова в предложении (нужен для вывода предложения без разделителей в начале);
- int is_final — элеменет, в котором хранится информация о том, является ли предложение финальным;
- int sent_len — длина предложения в символах;
- int words_count — количество слов в предложении.

Элементы структуры Text:

- struct Sentence *sentences — указатель на первый элемент массива предложений (текста);

- int text_len — количество предложений в тексте.

Непосредственно считывание текста производится при помощи функций struct Sentence get_sentence() и struct Text get_text().

Функция get_sentence возвращает тип данных struct Sentence и ничего не принимает на вход. Считывание предложения производится посимвольно при помощи стандартной функции fgetwc в цикле do while до тех пор, пока символ не будет равен символу, которым завершается ввод, хранящемуся в именованной константе EOINP, по умолчанию этот символ равен „\0“ (для ввода этого символа с клавиатуры в ОС Ubuntu используется комбинация клавиш Ctrl+Shift+2) или точке. В цикле также предусмотрено расширение блока памяти, выделенного для хранения предложения, если количества уже выделенной памяти недостаточно. После цикла в конец предложения записывается символ конца строки, после чего всем элементам возвращаемой структуры присваиваются определённые значения. Далее функция возвращает полученную структуру и завершает свою работу.

Функция get_text возвращает тип данных struct Text и ничего не принимает на вход. Считывание текста происходит в цикле, каждое новое предложение считывается при помощи функции get_sentence после считывания предложения производится его обработка при помощи функции split (принцип её действия будет описан позже). После обработки предложения производится проверка, встречалось ли предложение ранее, сравнение предложений посимвольно без учёта регистра при помощи стандартной функции wcscasecmp, которая возвращает 0 если предложения равны. Если предложение встречалось ранее, то память, выделенная для хранения информации о нём освобождается и далее это предложение нигде не учитывается. Если значение элемента is_final очередного предложения не равно нулю, то цикл прерывается. Как и в функции get_sentence предусмотрено расширение блока памяти, хранящего указатели на структуры предложений. Функция возвращает полученную структуру text.

2.2 Функции обработки текста

Для разделения функции на слова и получения индекса первого слова используется функция split.

Функция void split(struct Sentence *sent) ничего не возвращает, а на вход принимает адрес структуры, которую необходимо изменить. Разделение предложения на слова происходит при помощи стандартной функции wcstok, которая изменяет исходную строку (предложение) из-за чего теряется информация о некоторых разделителях между словами. Поэтому необходимо создать копию предложения для хранения разделителей. Это реализовано при помощи стандартной функции wcscpy. Далее выделяется память для хранения указателей на слова, после чего в цикле while инициализируются значения указателей и определяется количество слов:

```
int i = 0;
word = wcstok(sent_copy, L", .\n\t", &tmp);
if (word != NULL) first_word = sent->sentence + (word - sent_copy);
else first_word = sent->sentence + wcslen(sent->sentence) - 1;
while (word != NULL) {
    words[i++] = word;
    word = wcstok(NULL, L", .\n\t", &tmp);
}
```

Далее полученные значение присваиваются соответствующим элементам переданной структуры:

```
sent->sent_copy = sent_copy;
sent->words = words;
sent->words_count = i;
sent->first_word = first_word;
```

После присваивания функция завершает свою работу.

Выполнение первой подзадачи (раскрашивание текста) производится при выводе, как и выполнение второй и четвёртой задач.

Сортировка текста, необходимая для решения третьей подзадачи производится при помощи встроенной функции qsort:

```
qsort(text.sentences, text.text_len, sizeof(struct Sentence), last_words_cmp);
```

Функция-компаратор int last_words_cmp(const void *a, const void *b) оперирует с адресами на переменные типа struct Sentence, поэтому в теле функции необходимо привести тип const void к типу struct Sentence*:

```
struct Sentence *sent1 = (struct Sentence *) a;
struct Sentence *sent2 = (struct Sentence *) b;
```

Далее необходимо получить получить адрес последнего слова в предложении:

```
wchar_t *last_word1 = sent1->words[sent1->words_count - 1];
wchar_t *last_word2 = sent2->words[sent2->words_count - 1];
```

Далее функция возвращает результат сравнения последних слов.

2.3 Функции вывода текста

Для реализации первой подзадачи (вывода цветного текста) используется функция void put_printed_sent(struct Sentence sent). Для работы этой функции определяются следующие именованные константы:

```
#define RESET_C L"\033[0m"
#define RED L"\033[1;91m"
#define BLUE L"\033[1;94m"
#define GREEN L"\033[1;92m"
#define YELLOW L"\033[1;93m"
```

Функция принимает на вход структуру Sentence, далее, если количество слов в предложении больше нуля в цикле for выводит посимвольно на экран все символы разделители, находящиеся до текущего слова в предложении, потом производится цветной вывод слова в зависимости от остатка при делении его длины на 3, это делается при помощи оператора Switch:

```
switch (wcslen(sent.words[j]) % 3) {
     case 0:
          wprintf(RED L"%ls" RESET_C, sent.words[j]);
          break;
     case 1:
          wprintf(BLUE L"%ls" RESET_C, sent.words[j]);
          break;
     case 2:
          wprintf(GREEN L"%ls" RESET_C, sent.words[j]);
          break;
     case 3:
          wprintf(YELLOW L"%ls" RESET_C, sent.words[j]);
          break;
```

После вывода всех слов производится вывод оставшихся в предложении символов-разделителей.

Для реализации второй подзадачи реализована функция int put_start_end_upper_words(struct Sentence sent). Данная функция возвращает значение типа int — знак того, что в предложении есть (или нет) подходящие слова. Основу функции составляет цикл for, в котором рассматриваются все слова предложения и выводятся только те, которые начинаются и заканчиваются на заглавную букву:

```
for (int i = 0; i < sent.words_count; i++) {
     wchar_t *word = sent.words[i];
     if (iswupper(word[0]) && iswupper(word[wcslen(word) - 1])) {
          wprintf(L" %ls ", word);
          sign = 1;
     }
}
```

Функции, используемые для реализации третьей подзадачи описаны в разделе 2.2. При вызове этой (третьей) опции, пользователь должен учитывать, что далее программа будет оперировать только с отсортированным текстом.

Для реализации четвёртой подзадачи реализована функция void put_sent_without_numbers(struct Sentence sent). Реализация функции похожа на реализацию функции put_printed_sent, за исключением того, что в ней используется не оператор switch, а оператор if, в котором проверяется, является ли слово числом (с помощью стандартной функции wcstoll). Если слово является числом, то оно не выводится.

2.4 Функция main

Для корректной работы с кириллическими символами в функции main используется функция setlocale:

```
setlocale(LC_ALL, "");
```

Далее на консоль выводится контекстное меню для пользователя, которое для удобства чтения кода выделено в отдельную функцию void print_menu().

Для хранения текста создаётся переменная text типа struct Text, после чего выводится подсказка для пользователя: «Введите текст, который хотите обработать:». При помощи функции get_text введённый пользователем текст считывается в переменную text, после чего производится очистка буфера ввода (в нём остался перевод строки) при помощи функции fgetwc.

После считывания текста программа начинает выполнять цикл while, нужный для того чтобы пользователь мог выбираться несколько опций обработки подряд и выходить из программы тогда, когда ему это будет нужно. В цикле выводится сообщение для пользователя: «Пожалуйста, введите номер желаемой опции обработки текста:». Введённая цифра считывается в переменную int oprtion, после чего производится очистка буфера.

Далее с помощью оператора switch определяется какую подзадачу нужно выполнить. Если введённый символ не соответствует ни одной опции выводится сообщение: «Неверно введён номер опции.».

Если пользователь выбрал первую опцию обработки, то с в цикле for производится вывод всех предложений функцией put_printed_sent (см. раздел 2.3).

Если пользователь выбрал вторую опцию, то выполняется следующий код:

```
for (int i = 1; i < text.text_len + 1; i++) {
     wprintf(L"В предложении %d:", i);
     if (!put_start_end_upper_words(text.sentences[i])){
          wprintf(RED L" подходящих слов не нашлось." RESET_C);
     }
     wprintf(L"\n");
}
```

Таким образом на консоль выводится строка вида «В предложении <номер предложения> : <подходящие слова>» если в предложении есть подходящие слова и строка вида «В предложении <номер предложения> : подходящих слов не нашлось.» если подходящий слов в предложении нет.

Если пользователь выбрал опцию под номером 3, то на консоль выводится отсортированный текст, каждое предложение с новой строки. Алгоритм сортировки см. в разделе 2.2.

Если пользователь выбрал четвёртую опцию то в цикле for производится вывод всех предложений без чисел при помощи функции put_sent_without_numbers (см. раздел 2.3).

После вывода обработанного текста выводится сообщение: «Пожалуйста, введите 1 для выбора другой опции обработки или 0 для выхода из программы:». Если пользователь введёт любой символ кроме нуля, выполнение программы продолжится.

После того, как пользователь решил совершить выход из программы производится очистка всей динамически выделенной памяти:

```
for (int i = 0; i < text.text_len; i++) {
        free(text.sentences[i].sentence);
        free(text.sentences[i].sent_copy);
        free(text.sentences[i].words);
    }
    free(text.sentences);
```

Строка *return 0;* завершает выполнение программы.

2.5 Разделение функций на файлы и создание Makefile

Вся программа разделена на следующие файлы:

- main.c — основной файл программы, в нём содержится функция main. С помощью #inlclude в него включены все остальные заголовочные файла;
- structures.h — содержит объявления структур;

- read_funcs.h — файл, содержащий объявления функций считывания (get_sentence, ger_text) и все необходимые инструкции для препроцессора (именованные константы, стандартные библиотеки), также в него включён файл structures.h;
- read_funcs.c — содержит определения функция из файла read_funcs.h;
- processing_funcs.h — содержит объявления функций split и last_word_cmp;
- processing_funcs.c — содержит определения функция из файла processing_funcs.h, также в него включён файл structures.h;
- output_funcs.h — содержит объявления функций put_printed_sent, put_sent_without_numbers, put_start_end_upper_words также в него включён файл structures.h;
- output_funcs.c — содержит определения функций из заголовочного файла output_funcs.h.

В makefile прописаны все цели для создания нужных объектных и исполняемого файлов а также все зависимости.

2.6 Требования по вводу и обработка исключительных случаев

Для корректной работы программы необходимо, чтобы пользователь закончил ввод текста символом «\0», в противном случае ввод будет продолжаться, пока не будет введён требуемый символ.

При вводе неправильного номера опции будет выведено сообщение об ошибке.

Если во время ввода предложение не будет закончено (не будет точки перед символом \0), то все разделители и прочие символы до «\0» также будут считаться частью предложения и будут учитываться при обработке, длина последнего слова в таком предложении будет равна длине последнего слова до «\0» или равна 0 если до «\0» были только разделители.

При выборе опции удаления чисел из предложения на консоль будет выведен текст без чисел (числа будут удалены), однако в исходном тексте, который хранится в программе числа будут сохранены, для того чтобы пользователь при желании мог раскрасить эти числа (с помощью первой опции) или учитывать их при сортировке предложений. Однако если пользователь однажды отсортирует текст, то далее программа будет оперировать только с отсортированным текстом.

Программа удаляет одинаковые предложения без учёта регистра (то есть визуально они могут отличаться) при этом в тексте будет оставлена самая первая версия такого предложения, например текст «Предложение. ПРЕДЛОЖЕНИЕ.» послед удаления будет выглядеть так: «Предложение.».

При выводе слов начинающихся и заканчивающихся на заглавную букву слово состоящее из одной заглавной буквы считается подходящим, такие слова также буду выведены.

Если в предложении встречаются символы перевода строки, то при выводе каждого предложения с новой строки (в третьей подзадаче), символы перевода строки, находящиеся внутри предложения также будут выведены.

ЗАКЛЮЧЕНИЕ

В результате данной курсовой работы были изучены основные принципы обработки строк на языке Си. Получены навыки использования структур, работы с динамической памятью и работы с функциями заголовочных файлов wchar.h и wctype.h.

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

Для достижения цели были выполнены задачи:

- Реализовано считывание текста в структуру с использованием динамической памяти;
- Написаны функции обработки текста в соответствии с заданием;
- Программа разбита на файлы и написан makefile для сборки программы;
- Произведена отладка программы (пример работы программы см. в приложении А.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

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

3. Cplusplus URL: http://cplusplus.com (дата обращения: 14.12.2020).

ПРИЛОЖЕНИЕ А
ПРИМЕР РАБОТЫ ПРОГРАММЫ

Вывод консоли после вызова утилиты make:

```
gcc -c output_funcs.c
gcc -c processing_funcs.c
gcc -c read_funcs.c
gcc -c main.c
gcc output_funcs.o processing_funcs.o read_funcs.o main.o
```

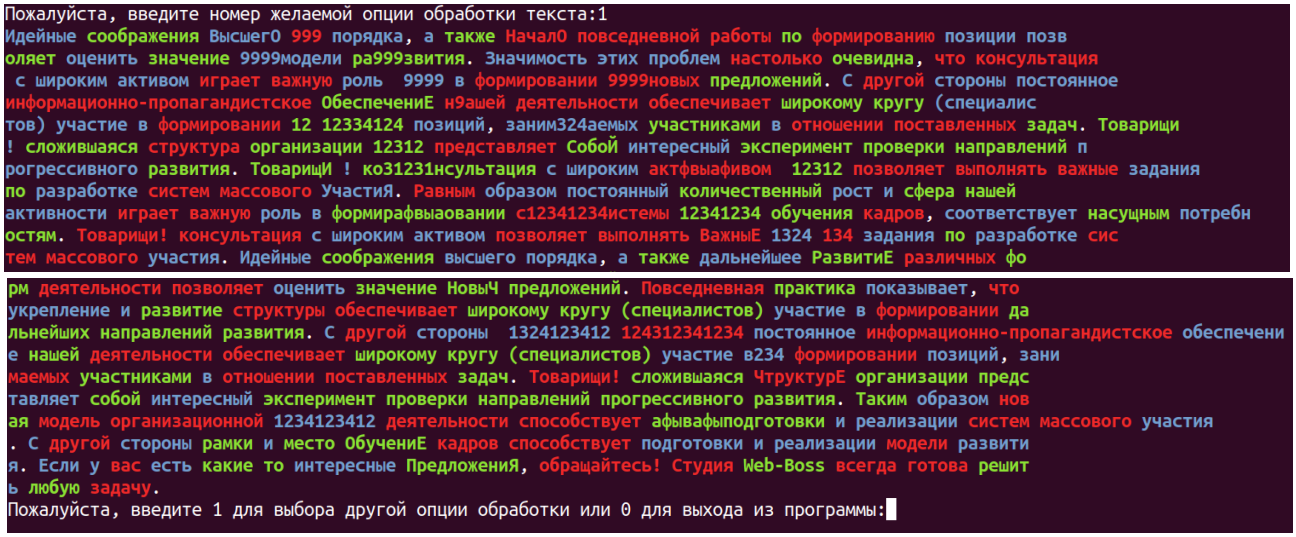
Вывод меню после запуска исполняемого файла:

```
Уважаемый пользователь, вас приветствует программа обработки текста!
Возможны следующие опции обработки текста:
1) Раскрасить текст.
2) Вывести слова, начинающиеся и заканчивающиеся на заглавную букву.
3) Отсортировать предложения по длине последнего слова в предложении.
4) Удалить все числа из предложений.
Введите текст, который хотите обработать:
```

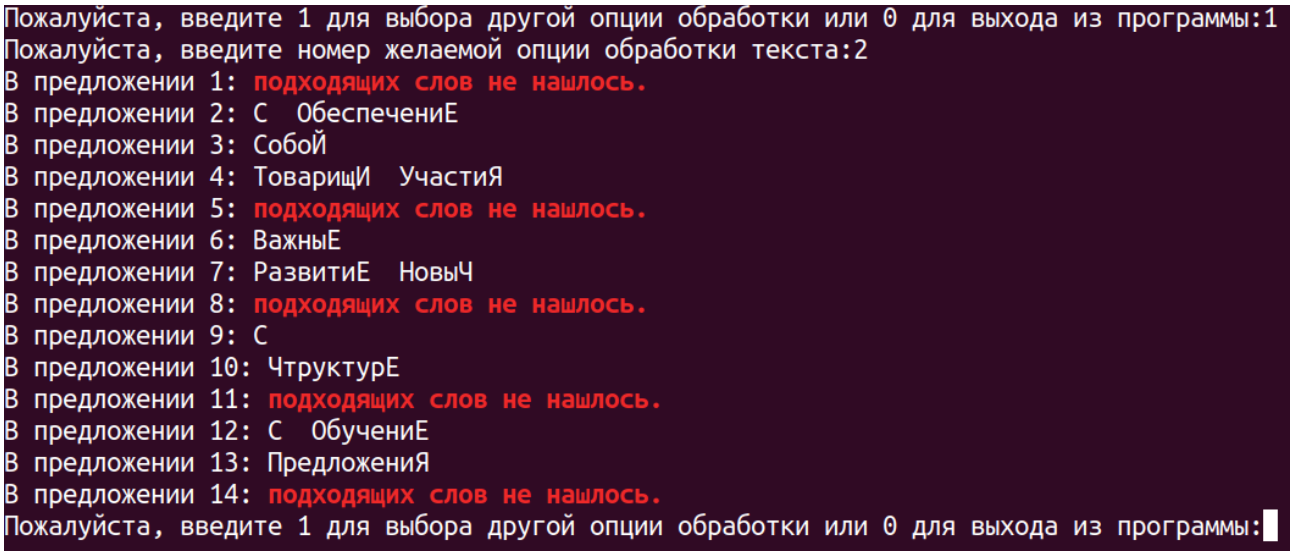
Произведён ввод текста, оканчивающийся символом «\0»:

```
Уважаемый пользователь, вас приветствует программа обработки текста!
Возможны следующие опции обработки текста:
1) Раскрасить текст.
2) Вывести слова, начинающиеся и заканчивающиеся на заглавную букву.
3) Отсортировать предложения по длине последнего слова в предложении.
4) Удалить все числа из предложений.
Введите текст, который хотите обработать:
Идейные соображения ВысшегО 999 порядка, а также НачалО повседневной работы по формированию позиции позв
оляет оценить значение 9999модели ра999звития. Значимость этих проблем настолько очевидна, что консультация
 с широким активом играет важную роль  9999 в формировании 9999новых предложений. С другой стороны постоянное
информационно-пропагандистское ОбеспечениЕ н9ашей деятельности обеспечивает широкому кругу (специалис
тов) участие в формировании 12 12334124 позиций, занимЗ24аемых участниками в отношении поставленных задач. Товарищи
! сложившаяся структура организации 12312 представляет СобоЙ интересный эксперимент проверки направлений п
рогрессивного развития. ТоварищИ ! ко31231нсультация с широким актфвыафивом  12312 позволяет выполнять важные задания
по разработке систем массового УчастиЯ. Равным образом постоянный количественный рост и сфера нашей
активности играет важную роль в формирафвыаовании с12341234истемы 12341234 обучения кадров, соответствует насущным потребн
остям. Товарищи! консультация с широким активом позволяет выполнять ВажныЕ 1324 134 задания по разработке сис
тем массового участия. Идейные соображения высшего порядка, а также дальнейшее РазвитиЕ различных фо
рм деятельности позволяет оценить значение НовыЧ предложений. Повседневная практика показывает, что
укрепление и развитие структуры обеспечивает широкому кругу (специалистов) участие в формировании да
льнейших направлений развития. С другой стороны  1324123412 124312341234 постоянное информационно-пропагандистское обеспечени
е нашей деятельности обеспечивает широкому кругу (специалистов) участие в234 формировании позиций, зани
маемых участниками в отношении поставленных задач. Товарищи! сложившаяся ЧтруктурЕ организации предс
тавляет собой интересный эксперимент проверки направлений прогрессивного развития. Таким образом нов
ая модель организационной 1234123412 деятельности способствует афывафыподготовки и реализации систем массового участия
. С другой стороны рамки и место ОбучениЕ кадров способствует подготовки и реализации модели развити
я. Если у вас есть какие то интересные ПредложенияЯ, обращайтесь! Студия Web-Boss всегда готова решит
ь любую задачу.^@
Пожалуйста, введите номер желаемой опции обработки текста:
```

Работа первой опции:



Работа второй опции:



Работа третьей опции:

```
Пожалуйста, введите 1 для выбора другой опции обработки или 0 для выхода из программы:1
Пожалуйста, введите номер желаемой опции обработки текста:3
С другой стороны рамки и место ОбучениЕ кадров способствует подготовки и реализации модели развити
я.
С другой стороны постоянное
информационно-пропагандистское ОбеспечениЕ н9ашей деятельности обеспечивает широкому кругу (специалис
тов) участие в формировании 12 12334124 позиций, заним324аемых участниками в отношении поставленных задач.
Равным образом постоянный количественный рост и сфера нашей
активности играет важную роль в формирафвыаовании с12341234истемы 12341234 обучения кадров, соответствует насущным потребн
остям.
С другой стороны  1324123412 124312341234 постоянное информационно-пропагандистское обеспечени
е нашей деятельности обеспечивает широкому кругу (специалистов) участие в234 формировании позиций, зани
маемых участниками в отношении поставленных задач.
Если у вас есть какие то интересные ПредложениЯ, обращайтесь! Студия Web-Boss всегда готова решит
ь любую задачу.
ТоварищИ ! ко31231нсультация с широким актфвыафивом  12312 позволяет выполнять важные задания
по разработке систем массового УчастиЯ.
Товарищи! консультация с широким активом позволяет выполнять ВажныЕ 1324 134 задания по разработке сис
тем массового участия.
Таким образом нов
ая модель организационной 1234123412 деятельности способствует афывафыподготовки и реализации систем массового участия
.
Товарищи
! сложившаяся структура организации 12312 представляет СобоЙ интересный эксперимент проверки направлений п
рогрессивного развития.
Повседневная практика показывает, что
укрепление и развитие структуры обеспечивает широкому кругу (специалистов) участие в формировании да
льнейших направлений развития.
Товарищи! сложившаяся ЧтруктурЕ организации предс
тавляет собой интересный эксперимент проверки направлений прогрессивного развития.
Идейные соображения ВысшегО 999 порядка, а также НачалО повседневной работы по формированию позиции позв
оляет оценить значение 9999модели ра999звития.
Значимость этих проблем настолько очевидна, что консультация
 с широким активом играет важную роль  9999 в формировании 9999новых предложений.
Идейные соображения высшего порядка, а также дальнейшее РазвитиЕ различных фо
рм деятельности позволяет оценить значение НовыЧ предложений.
Пожалуйста, введите 1 для выбора другой опции обработки или 0 для выхода из программы:
```

Работа четвёртой опции:

```
Пожалуйста, введите 1 для выбора другой опции обработки или 0 для выхода из программы:1
Пожалуйста, введите номер желаемой опции обработки текста:4
 С другой стороны рамки и место ОбучениЕ кадров способствует подготовки и реализации модели развити
я. С другой стороны постоянное
информационно-пропагандистское ОбеспечениЕ н9ашей деятельности обеспечивает широкому кругу (специалис
тов) участие в формировании  позиций, заним324аемых участниками в отношении поставленных задач. Равным образом постоянный количественный рост и сфера нашей
активности играет важную роль в формирафвыаовании с12341234истемы  обучения кадров, соответствует насущным потребн
остям. С другой стороны  постоянное информационно-пропагандистское обеспечени
е нашей деятельности обеспечивает широкому кругу (специалистов) участие в234 формировании позиций, зани
маемых участниками в отношении поставленных задач. Если у вас есть какие то интересные ПредложениЯ, обращайтесь! Студия Web-Boss всегда готова решит
ь любую задачу. ТоварищИ ! ко31231нсультация с широким актфвыафивом  позволяет выполнять важные задания
по разработке систем массового УчастиЯ. Товарищи! консультация с широким активом позволяет выполнять ВажныЕ  задания по разработке сис
тем массового участия. Таким образом нов
ая модель организационной  деятельности способствует афывафыподготовки и реализации систем массового участия
. Товарищи
! сложившаяся структура организации  представляет СобоЙ интересный эксперимент проверки направлений п
рогрессивного развития. Повседневная практика показывает, что
укрепление и развитие структуры обеспечивает широкому кругу (специалистов) участие в формировании да
льнейших направлений развития. Товарищи! сложившаяся ЧтруктурЕ организации предс
тавляет собой интересный эксперимент проверки направлений прогрессивного развития.Идейные соображения ВысшегО  порядка, а также НачалО повседневной работы по формированию позиции позв
оляет оценить значение 9999модели ра999звития. Значимость этих проблем настолько очевидна, что консультация
 с широким активом играет важную роль  в формировании 9999новых предложений. Идейные соображения высшего порядка, а также дальнейшее РазвитиЕ различных фо
рм деятельности позволяет оценить значение НовыЧ предложений.
Пожалуйста, введите 1 для выбора другой опции обработки или 0 для выхода из программы:0
```

Пример обработки неправильно введённого номера опции:

```
Пожалуйста, введите 1 для выбора другой опции обработки или 0 для выхода из программы:1
Пожалуйста, введите номер желаемой опции обработки текста:7
Неверно введён номер опции.
Пожалуйста, введите 1 для выбора другой опции обработки или 0 для выхода из программы:1
Пожалуйста, введите номер желаемой опции обработки текста:8
Неверно введён номер опции.
Пожалуйста, введите 1 для выбора другой опции обработки или 0 для выхода из программы:1
Пожалуйста, введите номер желаемой опции обработки текста:]
Неверно введён номер опции.
Пожалуйста, введите 1 для выбора другой опции обработки или 0 для выхода из программы:1
Пожалуйста, введите номер желаемой опции обработки текста:9
Неверно введён номер опции.
Пожалуйста, введите 1 для выбора другой опции обработки или 0 для выхода из программы:1
Пожалуйста, введите номер желаемой опции обработки текста:
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл structures.h:

```
#pragma once

struct Sentence {
    wchar_t *sentence;
    wchar_t *sent_copy;
    wchar_t **words;
    wchar_t *first_word;

    int is_final;
    int sent_len;
    int words_count;
};

struct Text {
    struct Sentence *sentences;
    int text_len;
};
```

Файл main.c:

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

struct Sentence {
    wchar_t *sentence;
    wchar_t *sent_copy;
    wchar_t **words;
    wchar_t *first_word;

    int is_final;
    int sent_len;
    int words_count;
};

struct Text {
    struct Sentence *sentences;
    int text_len;
};h>
#include <locale.h>

#include "output_funcs.h"
#include "processing_funcs.h"
#include "read_funcs.h"
#include "structures.h"

#define RESET_C L"\033[0m"
#define RED L"\033[1;91m"
```

```
#define TRUE 1

void print_menu();

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

    print_menu();

    struct Text text;
     wprintf(L"Введите текст, который хотите обработать: \n");
    text = get_text();
    fgetwc(stdin);

    int option;
    int exit;
    while (TRUE){
     wprintf(L"Пожалуйста, введите номер желаемой опции обработки
текста:");
          wscanf(L"%d", &option);
          fgetwc(stdin);
          switch (option) {
          case 1:
              for (int i = 0; i < text.text_len; i++) {
                  put_printed_sent(text.sentences[i]);
              }
              wprintf(L"\n");
              break;
          case 2:
              for (int i = 1; i < text.text_len + 1; i++) {
                  wprintf(L"В предложении %d:", i);
                  if
(!put_start_end_upper_words(text.sentences[i])){
                    wprintf(RED L" подходящих слов не нашлось."
RESET_C);
                  }
                  wprintf(L"\n");
              }
              break;
          case 3:
              qsort(text.sentences, text.text_len, sizeof(struct
Sentence), last_words_cmp);
              for (int i = 0; i < text.text_len; i++) {
                  fputws(text.sentences[i].first_word, stdout);
                  wprintf(L"\n");
              }
              break;
          case 4:
              for (int i = 0; i < text.text_len; i++) {
                  put_sent_without_numbers(text.sentences[i]);
              }
              wprintf(L"\n");
              break;
```

```
            default:
                wprintf(L"Неверно введён номер опции.\n");
        }
        wprintf(L"Пожалуйста, введите 1 для выбора другой опции
обработки или 0 для выхода из программы:");
        wscanf(L"%d", &exit);
        if (!exit) break;
    }

    for (int i = 0; i < text.text_len; i++) {
        free(text.sentences[i].sentence);
        free(text.sentences[i].sent_copy);
        free(text.sentences[i].words);
    }
    free(text.sentences);

    return 0;
}

void print_menu(){
    wprintf(L"Уважаемый пользователь, вас приветствует программа
обработки текста!\n"\
    "Возможны следующие опции обработки текста:\n"\
    "1) Раскрасить текст.\n2) Вывести слова, начинающиеся и
заканчивающиеся на заглавную букву.\n"\
    "3) Отсортировать предложения по длине последнего слова в
предложении.\n"\
    "4) Удалить все числа из предложений.\n");
}
```

Файл read_funcs.h:

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

#include "structures.h"
#include "processing_funcs.h"

#define TRUE 1
#define FALSE 0
#define SENT_STEP 50
#define TEXT_STEP 5
#define EOINP L'\0'

struct Sentence get_sentence();

struct Text get_text();
```

Файл read_funcs.c:

```
#include "read_funcs.h"

struct Sentence get_sentence() {
    struct Sentence sent;
    int sent_mem_size = SENT_STEP;
    int sent_len = 0;
    int is_final = FALSE;

    sent.sentence = calloc(sent_mem_size, sizeof(wchar_t));

    wchar_t sym;
    do {
        sym = (wchar_t) fgetwc(stdin);
        if (sym == EOINP){
            is_final = TRUE;
            break;
        }
        sent.sentence[sent_len++] = sym;
        if (sent_len == sent_mem_size - 2) {
            sent_mem_size += SENT_STEP;
            sent.sentence = realloc(sent.sentence, sent_mem_size *
(sizeof(wchar_t)));
        }
    } while (sym != L'.');
    sent.sentence[sent_len] = L'\0';

    sent.sent_len = sent_len;
    sent.is_final = is_final;
    return sent;
}

struct Text get_text() {
    int text_mem_size = TEXT_STEP;
    struct Text text;
    int text_len = 0;

    text.sentences = calloc(text_mem_size, sizeof(struct
Sentence));

    while (TRUE) {
        struct Sentence sentence = get_sentence();
        split(&sentence);

        int i;
        for (i = 0; i < text_len; i++) {
            if (!(wcscasecmp(text.sentences[i].first_word,
sentence.first_word))) break;
        }
        if (i == text_len) {
            if (sentence.sent_len) text.sentences[text_len++] =
sentence;
        }
        else{
```

```
            free(sentence.sentence);
            free(sentence.words);
            free(sentence.sent_copy);
        }

        if (sentence.is_final) break;

        if (text_len == text_mem_size - 2) {
            text_mem_size += TEXT_STEP;
            text.sentences = realloc(text.sentences, text_mem_size
* (sizeof(struct Sentence)));
        }
    }

    text.text_len = text_len;
    return text;
}
```

Файл processing_funcs.c:

```
#pragma once

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

#include "structures.h"

#define TRUE 1
#define FALSE 0

void split(struct Sentence *sent);
int last_words_cmp(const void *a, const void *b);
```

Файл processing_funcs.c:

```
#include "processing_funcs.h"

void split(struct Sentence *sent) {
    wchar_t *sent_copy = calloc(sent->sent_len, (sizeof(wchar_t))
+ 1);
    wcscpy(sent_copy, sent->sentence);

    wchar_t **words = calloc(sent->sent_len, sizeof(wchar_t *));
    wchar_t *word;
    wchar_t *first_word;
    wchar_t *tmp;

    int i = 0;
    word = wcstok(sent_copy, L", .\n\t", &tmp);
    if (word != NULL) first_word = sent->sentence + (word -
sent_copy);
```

```
    else first_word = sent->sentence + wcslen(sent->sentence) - 1;
    while (word != NULL) {
        words[i++] = word;
        word = wcstok(NULL, L", .\n\t", &tmp);
    }

    sent->sent_copy = sent_copy;
    sent->words = words;
    sent->words_count = i;
    sent->first_word = first_word;
}

int last_words_cmp(const void *a, const void *b) {
    struct Sentence *sent1 = (struct Sentence *) a;
    struct Sentence *sent2 = (struct Sentence *) b;
    wchar_t *last_word1 = sent1->words[sent1->words_count - 1];
    wchar_t *last_word2 = sent2->words[sent2->words_count - 1];
    if (sent1->words_count > 0 && sent2->words_count > 0) {
        return wcslen(last_word1) - wcslen(last_word2);
    } else if (sent1->words_count > 0){
        return wcslen(last_word1);
    } else if (sent2->words_count > 0){
        return -wcslen(last_word2);
    } else return 0;
}
```

Файл output_funcs.h:

```
#pragma once

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

#include "structures.h"

#define TRUE 1
#define FALSE 0
#define SENT_STEP 50
#define TEXT_STEP 5
#define EOINP L'\0'

#define RESET_C L"\033[0m"
#define RED L"\033[1;91m"
#define BLUE L"\033[1;94m"
#define GREEN L"\033[1;92m"
#define YELLOW L"\033[1;93m"

void put_printed_sent(struct Sentence sent);
void put_sent_without_numbers(struct Sentence sent);
int put_start_end_upper_words(struct Sentence sent);
```

Файл output_funcs.c:

```
#include "output_funcs.h"

void put_printed_sent(struct Sentence sent) {
    if (sent.words_count) {
        int i, j;
        for (i = 0, j = 0; j < sent.words_count; i++) {
            while (wcschr(L" ,\t\n", sent.sentence[i])) {
                wprintf(L"%lc", sent.sentence[i++]); //
            }
            switch (wcslen(sent.words[j]) % 3) {
                case 0:
                    wprintf(RED L"%ls" RESET_C, sent.words[j]);
                    break;
                case 1:
                    wprintf(BLUE L"%ls" RESET_C, sent.words[j]);
                    break;
                case 2:
                    wprintf(GREEN L"%ls" RESET_C, sent.words[j]);
                    break;
                case 3:
                    wprintf(YELLOW L"%ls" RESET_C, sent.words[j]);
                    break;
            }
            i += (int) wcslen(sent.words[j++]) - 1;
        }
        for (; i < sent.sent_len; i++) {
            wprintf(L"%lc", sent.sentence[i]);
        }
    } else wprintf(L"%ls", sent.sentence);
}

int put_start_end_upper_words(struct Sentence sent) {
    int sign = 0;
    for (int i = 0; i < sent.words_count; i++) {
        wchar_t *word = sent.words[i];
        if (iswupper(word[0]) && iswupper(word[wcslen(word) - 1]))
{
            wprintf(L" %ls ", word);
            sign = 1;
        }
    }
    return sign;
}

void put_sent_without_numbers(struct Sentence sent) {
    if (sent.words_count > 0) {
        int i, j;
        for (i = 0, j = 0; j < sent.words_count; i++) {
            while (wcschr(L" ,\t\n", sent.sentence[i])) {
                wprintf(L"%lc", sent.sentence[i++]); //
```

```
            }
            wchar_t *sign;
            wcstoll(sent.words[j], &sign, 10);
            if (*sign) {
                wprintf(L"%ls", sent.words[j]);
            }
            i += (int) wcslen(sent.words[j++]) - 1;
        }
        for (; i < sent.sent_len; i++) {
            wprintf(L"%lc", sent.sentence[i]);
        }
    } else wprintf(L"%ls", sent.sentence);
}
```

Makefile:

```
all: output_funcs.o processing_funcs.o read_funcs.o main.o
     gcc output_funcs.o processing_funcs.o read_funcs.o main.o -o
main
main.o: main.c output_funcs.h processing_funcs.h read_funcs.h
structures.h
     gcc -c main.c
output_funcs.o: output_funcs.c output_funcs.h structures.h
     gcc -c output_funcs.c
processing_funcs.o: processing_funcs.c processing_funcs.h
structures.h
     gcc -c processing_funcs.c
read_funcs.o: read_funcs.c read_funcs.h processing_funcs.h
structures.h
     gcc -c read_funcs.c
clean:
     rm *.o
```