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.

Студентка гр. 0382 _____ Рубежова Н.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студентка Рубежова Н.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) Сделать транслитерацию всех кириллических символов в тексте. Например, подстрока “Какой nice пень” должна принять вид “Kakoj nice pen’” (использовать ГОСТ 7.79-2000)
2) Для каждого предложения вывести все специальные символы в порядке уменьшения их кода.

3) Заменить все цифры в тексте их двоичным кодом.

4) Удалить все предложения в которых есть нечетные цифры.

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.

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 24.11.2020

Дата сдачи реферата: 29.12.2020

Дата защиты реферата: 29.12.2020

Студентка _____ Рубежова Н.А.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В ходе выполнения курсовой работы была реализована программа на языке Си, которая получает на вход текст и обрабатывает его с использованием функций стандартных библиотек. Ввод текста осуществляется до тех пор, пока не встретятся два переноса строки подряд. Между словами может быть либо пробел, либо запятая. Предложения разделены только точкой. Для хранения текста были использованы структуры Text, Sentence, Word. Также был организован выбор команды пользователем для обработки текста. Для сборки программы используется Makefile.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В курсовой работе реализована программа по обработке текста в зависимости от введенной пользователем команды. Возможность выбора команды реализована с помощью оператора множественного выбора switch. Для хранения текста использованы структуры Text, Sentence, Word, память на хранение текста выделяется динамически, программа поддерживает обработку текстов как с латинскими, так и с кириллическими символами, используя возможности библиотеки wchar.h.

Программа была разработана на операционной системе Linux Ubuntu в интерактивной среде разработки Clion и текстовом редакторе Vim. Компиляция и линковка осуществлялась с помощью Makefile.

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цель: Освоить принципы обработки строк и работы с символьными массивами в языке Си, а также реализовать программу для обработки текста, чтобы закрепить полученные знания на практике.

Для достижения цели были поставлены следующие задачи:

- Реализовать считывание текста;
- Реализовать функции по обработке текста:
- Обеспечить возможность выбора команды с помощью оператора switch;
- Верно создать Makefile.

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Считывание и хранение текста:

Для хранения текста использованы структуры Word, Sentence и Text.

Поля структуры Word:

- wchar_t* buf — изначально указатель на начало слова, после выделения памяти – си-строка, хранящая слово;
- int leng_word — длина слова.

Поля структуры Sentence:

- Word* buf_words — после выделения памяти хранит массив структур Word
- int leng_sent — количество слов в предложении
- wchar_t* str_sent – после выделения памяти хранит всё предложение целиком как си-строку
- int leng_str – длина предложения-строки с учетом нуль-терминатора
- int* buf_codes_symb – после выделения памяти хранит массив целочисленных кодов специальных символов этого предложения
- int leng_buf_codes_symb – длина массива кодов специальных символов предложения

Поля структуры Text:

- Sentence* buf_sents — указатель на массив структур Sentence;
- int leng_text — количество предложений в тексте.

Текст будем хранить в переменной inp, которая является объектом структуры Text. Ввод текста происходит посимвольно, первый символ считывается в функции main, далее передаем этот символ одной из функций ввода текста get_Text() вместе с адресом нашего объекта inp, чтобы при дальнейшем вводе мы могли обращаться и изменять поля структуры Text. Следующие функции ввода будут организованы вложенно.

Заходим в функцию get_Text(), которая в цикле будет вызывать функцию get_Sent() с целью присвоить i-ому элементу буфера предложений(который является полем структуры Text) структуру Sentence. А get_Sent() внутри себя вызывает еще одну вложенную функцию get_Word(), с целью присвоить i-ому элементу буфера слов(который является полем структуры Sentence) структуру Word. Таким образом, получилась иерархия вложенных функций.

С первым введенным символом программа «спускается» до get_Word(), в этой функции заполняет буфер, хранящий слово(вводя новые символы, пока не встретится разделитель слова – пробел или точка). По мере заполнения буфера счетчик считает длину слова leng_word. Как только буфер заполнится, его можно передать полю структуры Sentence – buf_words, там слово будет хранится целиком и являться элементом этого массива. Слова будут считываться(get_Word() вызываться), пока не встретится точка- символ конца предложения.Таким образом, Sentence заполняет свой буфер слов и от счетчика получит количество этих слов.

Аналогично Text, на каждой итерации вызывая функцию get_Sent(), заполняет свой массив предложений, пока не встретятся два переноса строки подряд(именно для этого нам нужен был цикл while). Полученный буфер предложений записывается в поле структуры Text.buf_sents, а также количество предложений в поле Text.leng_text.
При каждом создании вспомогательного массива мы выделяем память и не забываем освобождать ее после выполнения функции.

2.2. Решение подзадачи №1.

Нам необходимо транслитерировать кириллические символы текста. Для того, чтобы перебирать все символы текста воспользуемся двумя вложенными циклами for, чтобы обратиться к структурам Sentence и к их

полям str_sent, хранящим строку предложение. И уже в каждой такой строке мы будем перебирать символы, как i-ый элемент си-строки/массива.

Необходимо проверить является ли символ кириллическим. Поэтому запишем все кириллические символы в два массива: в одном будут храниться кириллические символы верхнего регистра, во втором- нижнего регистра. Помимо них создадим два массива соответствующих им латинских символов (верхнего и нижнего регистра) по ГОСТу 7.79-2000, опираясь на кодировку Unicode.

Будем перебирать все символы текста, воспользовавшись полями структур, и с помощью функции wcschr() проверим, является ли символ кириллическим, то есть входит в один из кириллических массивов. Если да, то функция возвращает указатель на кириллический символ массива, если нет – возвращает NULL. Указатель на кириллический символ в массиве поможет нам, используя арифметику указателей, выбрать элемент с таким же индексом соответствующего ему элемента из «транслитерированного» массива. Соответственно, разыменовав итоговый указатель, мы получим символ, на который нам и нужно заменить наш символ. Через поля структуры обращаемся к исходному символу и присваиваем новое значение – символ, полученный разыменования указателя.

Таким образом мы переберем все элементы текста и заменим их на соответствующие им транслитерированные символы.

2.3. Решение подзадачи №2.

Решение данной подзадачи реализовано в функции task_2(), которая принимает на вход указатель на объект структуры Text.

Во вложенном цикле проверяется каждый символ текста, является ли он специальным символом(аналогично первому пункту – создаем массив специальных символов и проверяем с помощью wcschr()). Если функция

выводит не NULL, то есть символ является специальным, то переменной int code присваиваем наш символ, теперь в code будет храниться код исходного символа и в предварительно выделенный буфер buf_ints запишем значение code. Перебирая все символы предложения, мы заполним наш буфер кодов специальных символов, присутствующих в предложении. Этот буфер передадим полю предложения buf_code_symb.

Чтобы отсортировать коды специальных символов воспользуемся функцией qsort(), которая умеет сортировать массивы любых типов, для этого нам понадобится сам массив, количество его элементов и функция, согласно которой будет происходить сортировка.

Для этого определим свою функцию-компаратор compare(), в которую будут передаваться указатели двух элементов массива. Так как нам нужно отсортировать массив в порядке уменьшения кода, то когда первый элемент будет больше второго, наш компаратор будет выводить – -1, если равны – 0, если второй больше первого – 1. Таким образом, по возвращаемым компаратором значениям функция qsort(), отсортирует коды специальных символов.

Осталось вывести их на экран. Для этого мы будем перебирать элементы уже отсортированного массива и выводить их через wprintf(), используя спецификатор “%lc”, ведь в таком случае по коду символа будет выводиться сам символ. Таким образом, для каждого предложения будет выведена последовательность специальных символов, расположенных в порядке уменьшения их кода.

2.4. Решение подзадачи №3.

Аналогично 1ому и 2ому пункту перебираем все символы текста и проверяем с помощью wcschr() , являются ли они цифрами из строки-массива «23456789», так как двоичное представление чисел 0 и 1 не отличается от десятичного. Если функция возвращает не NULL, то проверяем,

разыменовывая указатель, символ является 2 или 3? Вообще условно разделим интересующие нас цифры на 3 группы: 2 и 3 в бинарном виде занимают 2 символа, 4-7 – занимают 3 символа, 8 и 9 – по 4 символа. Соответственно, чтобы перезаписать один символ в строке на 2,3 или 4 нам придется расширять память с помощью функции realloc() на 1,2 и 3 sizeof(wchar_t) соответственно, а также сдвинуть символы, идущие после цифры на 1,2 и 3 позиции вперед соответветственно. После этого мы заполняем образовавшиеся «пустоты» единицами и нулями, в зависимости от того, какая цифра нам попалась. Таким образом, мы заменим все цифры в тексте на их двоичный код.

2.5. Решение подзадачи №4.

В подзадаче нам нужно удалить все предложения, ,в которых есть нечетные цифры.
Для этого, обращаясь к полям структур и используя вложенные циклы, переберем все символы текста. Если очередной символ оказывается цифрой 1,3,5 или 7, то мы должны удалить это предложение. Реализовывать мы это будем с помощью сдвига массива. Будем перебирать все структуры Sentence от 0 до text.leng_text. Если встречаем одну из неудовлетворяющих нас цифр, то в зависимости от того, какое это по счету предложение, и обращаясь к полям структур выполняем сдвиг: text_p->buf_sents[k]=text_p->buf_sents[k+1]. Но не забудем так же про leng_sent, ведь мы перебираем предложения и, удалив предложение, при переборе можем выйти за границу массива. Поэтому не забываем параллельно выполнять text_p->leng_text-=1.
Таким образом, мы переберем все символы текста, удалим неудовлетворяющие условию предложения, и удалим их, обращаясь к полям структур.

2.6. Вывод обработанного текста.

Вывод обработанного текста реализован в операторе множественного выбора switch.

Происходит ввод команды с консоли, пока пользователь не введет нулевую команду, которая ведет к завершению программы. На экран будет выводится подсказка для пользователя с описанием того, за какие действия какая команда отвечает.

Case 1 будет вести к выполнению первой подзадачи. Но так как в задании не указано вывести транслитерированный текст, текст будет только обрабатываться, а результат прогона программы пользователь может увидеть, введя команду 5,которая выводит хранящийся на данный момент текст.

Case 2 будет вести к выполнению второй подзадачи – выводу специальных символов для каждого предложения текста в порядке уменьшения их кодов. Для просмотра результата дополнительные команды вводить не нужно, результат выводится на экран.

Case 3 будет вести к выполнению третьей подзадачи, но так как в задании не указано выводить текст с замещенными на двоичный код цифрами, результат пользователь может посмотреть, введя команду 5..

Case 4 3 будет вести к выполнению четвертой подзадачи, но так как в задании указано лишь удалить предложения, в которых есть нечетные цифры, результат обработки текста пользователь может посмотреть, введя команду 5

Case 5 ведет к выводу текста, хранящемся в памяти на данный момент, обращаясь к полям структур.

При некорректном вводе команды, на экран выводится: Неверная команда. И предлагает ввести другой запрос.

3. ТЕСТИРОВАНИЕ

Вызов утилиты make:

```
programmer@vb:~/Desktop/cw$ make
gcc -c input_f.c
gcc -c task_1.c
gcc -c task_2.c
gcc -c task_3.c
gcc -c task_4.c
gcc -c menu.c
gcc input_f.c task_1.c task_2.c task_3.c task_4.c menu.c -o menu
programmer@vb:~/Desktop/cw$
```

Запуск исполняемого файла:

```
programmer@vb:~/Desktop/cw$ ./menu
Введите текст:
Примечание: между словами может быть пробел или запятая, предложения отделены то
чкой
```

Первая подзадача:

```
programmer@vb:~/Desktop/cw$ ./menu
Введите текст:
Примечание: между словами может быть пробел или запятая, предложения отделены точкой
K#ог$да мама просне4тся,тогда и будем зав2тракать.Холодно ночью.Завтра об0ещают !хор^ош$ую по3году.С утга он лю5би
л пить только колодезную воду,поэтому от сока отказался.Когда мама проснется,тогда и будем завтракать.Холодно ночь
ю.

Введите команду:
1 - сделать транслитерацию всех кириллических символов в тексте
2 - вывести для каждого предложения все специальные символы в порядке уменьшения их кода
3 - заменить все цифры в тексте их двоичным кодом
4 - удалить все предложения, в которых есть нечетные цифры
5 - вывести текст
0 - завершить программу
1
Введите следующую команду:
5
K#og$da mama prosne4tsâ,togda i budem zav2trakat'.Holodno noč'û.Zavtra ob0eŝaût !hor^oš$uû po3godu.S utra on lû5bi
l pit' tol'ko kolodeznuû vodu,poètomu ot soka otkazalsâ.Kogda mama prosnetsâ,togda i budem zavtrakat'.
Введите следующую команду:
```

Вторая подзадача:

```
programmer@vb:~/Desktop/cw$ ./menu
Введите текст:
Примечание: между словами может быть пробел или запятая, предложения отделены то
чкой
K#ог$да мама просне4тся,тогда и будем зав2тракать.Холодно ночью.Завтра об0ещают
!хор^ош$ую по3году.С утга он лю5бил пить только колодезную воду,поэтому от сока
отказался.Когда мама проснется,тогда и будем завтракать.Холодно ночью.

Введите команду:
1 - сделать транслитерацию всех кириллических символов в тексте
2 - вывести для каждого предложения все специальные символы в порядке уменьшения
 их кода
3 - заменить все цифры в тексте их двоичным кодом
4 - удалить все предложения, в которых есть нечетные цифры
5 - вывести текст
0 - завершить программу
2
Спецсимволы в порядке уменьшения кодов: $#  ^$!   Введите следующую команду:
```

Третья подзадача:

```
programmer@vb:~/Desktop/cw$ ./menu
Введите текст:
Примечание: между словами может быть пробел или запятая, предложения отделены то
чкой
K#ог$да мама просне4тся,тогда и будем зав2тракать.Холодно ночью.Завтра об0ещают
!хор^ош$ую по3году.С утга он лю5бил пить только колодезную воду,поэтому от сока
отказался.Когда мама проснется,тогда и будем завтракать.Холодно ночью.

Введите команду:
1 - сделать транслитерацию всех кириллических символов в тексте
2 - вывести для каждого предложения все специальные символы в порядке уменьшения
 их кода
3 - заменить все цифры в тексте их двоичным кодом
4 - удалить все предложения, в которых есть нечетные цифры
5 - вывести текст
0 - завершить программу
3
Введите следующую команду:
5
K#ог$да мама просне100тся,тогда и будем зав10тракать.Холодно ночью.Завтра об0еща
ют !хор^ош$ую по11году.С утга он лю101бил пить только колодезную воду,поэтому от
 сока отказался.Когда мама проснется,тогда и будем завтракать.
Введите следующую команду:
```

Четвертая подзадача:

```
programmer@vb:~/Desktop/cw$ ./menu
Введите текст:
Примечание: между словами может быть пробел или запятая, предложения отделены то
чкой
K#ог$да мама просне4тся,тогда и будем зав2тракать.Холодно ночью.Завтра об0ещают
!хор^ош$ую по3году.С утга он лю5бил пить только колодезную воду,поэтому от сока
отказался.Когда мама проснется,тогда и будем завтракать.Холодно ночью.

Введите команду:
1 - сделать транслитерацию всех кириллических символов в тексте
2 - вывести для каждого предложения все специальные символы в порядке уменьшения
 их кода
3 - заменить все цифры в тексте их двоичным кодом
4 - удалить все предложения, в которых есть нечетные цифры
5 - вывести текст
0 - завершить программу
4
Введите следующую команду:
5
K#ог$да мама просне4тся,тогда и будем зав2тракать.Холодно ночью.Когда мама просн
ется,тогда и будем завтракать.
Введите следующую команду:
```

ЗАКЛЮЧЕНИЕ

Разработана программа, которая обрабатывает поступающий на вход текст. Были использованы структуры, функции стандартных библиотек, оператор множественного выбора Switch. Была проделана работа по выделению динамической памяти, использованию такого типа данных, как wchar_t, освоены принципы работы с библиотекой <wchar.h>. Разработанная программа обрабатывает текст, согласно условиям задачи, в зависимости от команды, введенной пользователем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. http://www.cplusplus.com/

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: menu.c

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

#include "structs.h"
#include "input_f.h"
#include "task_1.h"
#include "task_2.h"
#include "task_3.h"
#include "task_4.h"

int main() {
    setlocale(LC_ALL,"");
    Text inp;
    wchar_t c=' ';
    wprintf(L"Введите текст:\nПримечание: между словами может быть пробел или запятая, предложения отделены точкой\n");
    c=getwchar();
    get_Text(c,&inp);
    int i;
    get_str_sent(&inp);
    for(i=0;i<inp.leng_text;i++){
        check_repeat(&(inp),&(inp.buf_sents[i]));
    }
    int com;
    wprintf(L"Введите команду:\n1 - сделать транслитерацию всех кириллических символов в тексте\n");
    wprintf(L"2 - вывести для каждого предложения все специальные символы в порядке уменьшения их кода\n");
    wprintf(L"3 - заменить все цифры в тексте их двоичным кодом\n");
```

```
    wprintf(L"4 - удалить все предложения, в которых есть нечетные
цифры\n");
    wprintf(L"5 - вывести текст\n");
    wprintf(L"0 - завершить программу\n");
    wscanf(L"%d",&com);
    while(com!=0){
        switch(com) {
            case 1:
                Task_1(&inp);
                break;
            case 2:
                Task_2(&inp);
                break;
            case 3:
                Task_3(&inp);
                break;
            case 4:
                Task_4(&inp);
                break;
            case 5:
                for (i = 0; i < inp.leng_text; i++) {
                    wprintf(L"%ls", inp.buf_sents[i].str_sent);
                }
                wprintf(L"\n");
                break;
            default:
                if (com != 0)wprintf(L"Неверная команда\n");
        }
        if(com!=0){
           wprintf(L"Введите следующую команду:\n");
           wscanf(L"%d",&com);
        }
    }
    return 0;
```

```
}
```

Файл: structs.h

```
#ifndef STRUCTS_H
#define STRUCTS_H

typedef struct {
    wchar_t* buf;
    int leng_word;
}Word;

typedef struct {
    Word* buf_words;
    int leng_sent;
    wchar_t* str_sent;
    int leng_str;//считает длину строки с нулевым символом в конце
    int* buf_codes_symb;
    int leng_buf_codes_symb;
}Sentence;

typedef struct {
    Sentence* buf_sents;
    int leng_text;
}Text;
#endif
```

Файл: input_f.c

```
#include <stdio.h>
#include <stdlib.h>
#include <wchar.h>
#include <locale.h>
#include "structs.h"
#include "input_f.h"

wchar_t get_Word(wchar_t c,Word* word_ptr) {
```

```
    wchar_t* tm;
    int size_buf=SIZE;
    wchar_t* buffer=(wchar_t*)calloc(size_buf,sizeof(wchar_t));
    int leng_w=0;
    if(c==' '||c==',')c=getwchar();
    while(c!=' '&&c!=','&&c!='.'){
        buffer[leng_w++]=c;
        if(leng_w==size_buf){
            size_buf*=2;
            tm=realloc(buffer,size_buf*sizeof(wchar_t));
            buffer=tm;
        }
        c=getwchar();
    }
    buffer[leng_w++]=c;
    buffer[leng_w]='\0';
    word_ptr->leng_word=leng_w;
    word_ptr->buf=(wchar_t*)malloc((leng_w+1)*sizeof(wchar_t));
    wcscpy(word_ptr->buf,buffer);
    free(buffer);
    return c;
}
wchar_t get_Sent(wchar_t sym, Sentence* sent_ptr){
    Word* tm;
    int count_words=SIZE;
    Word* buf_words_demo=(Word*)calloc(count_words,sizeof(Word));
    int leng_s=0;
    while(sym!='.'){
        sym=get_Word(sym,&buf_words_demo[leng_s++]);
        if(leng_s==count_words){
            count_words*=2;
            tm=realloc(buf_words_demo,count_words*sizeof(Word));
            buf_words_demo=tm;
        }
```

```
    }
    sent_ptr->leng_sent=leng_s;
    sent_ptr->buf_words=(Word*)malloc((leng_s+1)*sizeof(Word));
    int i;
    for(i=0;i<leng_s;i++){
        sent_ptr->buf_words[i]=buf_words_demo[i];
    }
    free(buf_words_demo);
    sym=getwchar();
    return sym;
}
void get_Text(wchar_t c, Text* inp_p){
    Sentence* tm;
    int count_sent=SIZE;
    Sentence*
buf_sents_demo=(Sentence*)malloc(count_sent*sizeof(Sentence));
    int leng_t=0;
    if(c=='\n') c=getwchar();
    while(c!='\n'){
        c=get_Sent(c,&buf_sents_demo[leng_t++]);
        if(leng_t==count_sent){
            count_sent*=2;
            tm=realloc(buf_sents_demo,count_sent*sizeof(Sentence));
            buf_sents_demo=tm;
        }
        if(c=='\n')c=getwchar();
    }
    inp_p->leng_text=leng_t;
    int i;
    inp_p->buf_sents=(Sentence*)malloc((leng_t+1)*sizeof(Sentence));
    for(i=0;i<leng_t;i++){
        inp_p->buf_sents[i]=buf_sents_demo[i];
    }
    free(buf_sents_demo);
```

```
}
void get_str_sent(Text* text_p){
    int i,j;
    for(i=0;i<text_p->leng_text;i++){
        int size=1;
        wchar_t* str=(wchar_t*)calloc(size,sizeof(wchar_t));
        for(j=0;j<text_p->buf_sents[i].leng_sent;j++){
            int size_add=text_p->buf_sents[i].buf_words[j].leng_word;
            str=realloc(str,(size+size_add)*sizeof(wchar_t));
            wcscat(str,text_p->buf_sents[i].buf_words[j].buf);
            size+=size_add;
        }

text_p->buf_sents[i].str_sent=(wchar_t*)malloc(size*sizeof(wchar_t));
        wcscpy(text_p->buf_sents[i].str_sent,str);
        text_p->buf_sents[i].leng_str=size;
        free(str);
    }
}
void check_repeat(Text* text_p,Sentence* sent_p){
    int i,j,k=0,flag;
    for(i=0;i<text_p->leng_text;i++){

if(wcscasecmp(sent_p->str_sent,text_p->buf_sents[i].str_sent)==0){
            flag=1;
            k++;
        }
        else
            flag=0;
        if(k>1&&flag==1){
            for(j=i;j<text_p->leng_text-1;j++){
                text_p->buf_sents[j]=text_p->buf_sents[j+1];
            }
```

```
            text_p->leng_text-=1;
        }
    }
}
```

Файл: input_f.h

```
#ifndef INPUT_F_H
#define INPUT_F_H

#define SIZE 50
wchar_t get_Word(wchar_t c,Word* word_ptr);
wchar_t get_Sent(wchar_t sym, Sentence* sent_ptr);
void get_Text(wchar_t c, Text* inp_p);
void get_str_sent(Text* text_p);
void check_repeat(Text* text_p,Sentence* sent_p);
#endif
```

Файл: task_1.c

```
#include <wchar.h>
#include <locale.h>
#include "structs.h"
#include "task_1.h"
void Task_1(Text* text_p){
    int i,j;
    wchar_t trans_up[]=L"ABVGDEËŽZIJKLMNOPRSTUFHCČŠŜ″Y′ÈÛÂ";
    wchar_t trans_low[]=L"abvgdeëžzijklmnoprstufhcčšŝ″y′èûâ";
    wchar_t cyr_up[]=L"АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";
    wchar_t cyr_low[]=L"абвгдеёжзийклмнопрстуфхцчшщъыьэюя";
    for(i=0;i<text_p->leng_text;i++){
        for(j=0;j<text_p->buf_sents[i].leng_str;j++){
            wchar_t c=text_p->buf_sents[i].str_sent[j];
            wchar_t* up=wcschr(cyr_up,c);
            if(up!=NULL) {
                text_p->buf_sents[i].str_sent[j]=*(up-
cyr_up+trans_up);
            }
```

```
            else{
                wchar_t* low = wcschr(cyr_low,c);
                if(low!=NULL){
                    text_p->buf_sents[i].str_sent[j]=*(low-
cyr_low+trans_low);
                }
            }

        }
    }
}
```

Файл: task_1.h

```
#ifndef UNTITLED_TASK_1_H
#define UNTITLED_TASK_1_H

void Task_1(Text* text_p);
#endif
```

Файл: task_2.c

```
#include <stdlib.h>
#include <wchar.h>
#include <locale.h>
#include "structs.h"
#include "task_2.h"
int compare(const void* a,const void* b){
    int temp_a=*(int*)a;
    int temp_b=*(int*)b;
    if(temp_a>temp_b)
        return -1;
    if(temp_a==temp_b)
        return 0;
    if(temp_a<temp_b)
        return 1;
}
void Task_2(Text* text_p){
```

```
    int i,j;
    for(i=0;i<text_p->leng_text;i++) {//перебираем предложения
        int*
buf_ints=(int*)calloc((text_p->buf_sents[i].leng_str),sizeof(int));
        int leng_buf_ints=0;
        for (j = 0; j < text_p->buf_sents[i].leng_str; j++)
{//перебираем символы внутри предложения
            wchar_t c = text_p->buf_sents[i].str_sent[j];
            wchar_t symbols[]=L"!\"#$%&'()*+-/:;<=>?@[\\]^_`{|}~";
            wchar_t* srch=wcschr(symbols,c);
            if(srch!=NULL){
                int code=*(srch);
                buf_ints[leng_buf_ints++]=code;
            }
        }

if(leng_buf_ints!=0)text_p->buf_sents[i].buf_codes_symb=(int*)malloc(l
eng_buf_ints*sizeof(int));
        for(j=0;j<leng_buf_ints;j++){
            text_p->buf_sents[i].buf_codes_symb[j]=buf_ints[j];
        }
        text_p->buf_sents[i].leng_buf_codes_symb=leng_buf_ints;
        free(buf_ints);
    }
    wprintf(L"Спецсимволы в порядке уменьшения кодов: ");
    for(i=0;i<text_p->leng_text;i++){
        if(text_p->buf_sents[i].leng_buf_codes_symb==0)wprintf(L"У
предложения спецсимволы отсуствуют. ");
        else {
            qsort(text_p->buf_sents[i].buf_codes_symb,
text_p->buf_sents[i].leng_buf_codes_symb, sizeof(int), compare);
            for (j = 0; j < text_p->buf_sents[i].leng_buf_codes_symb;
j++) {
                wprintf(L"%lc",
text_p->buf_sents[i].buf_codes_symb[j]);
            }
            wprintf(L" ");
```

```
        }
    }
}
```

Файл: task_2.h

```
#ifndef UNTITLED_TASK_2_H
#define UNTITLED_TASK_2_H

int compare(const void* a,const void* b);
void Task_2(Text* text_p);
#endif
```

Файл: task_3.c

```
#include <stdlib.h>
#include <wchar.h>
#include <locale.h>
#include "structs.h"
#include "task_3.h"
void Task_3(Text* text_p){
    int i,j,k;
    wchar_t numbers[]=L"0123456789";
    for(i=0;i<text_p->leng_text;i++){
        for(j=0;j<text_p->buf_sents[i].leng_str;j++){
            wchar_t c=text_p->buf_sents[i].str_sent[j];
            wchar_t* is_num=wcschr(numbers,c);
            if(is_num!=NULL) {
                if((*(is_num)=='2')||(*(is_num)=='3')){

text_p->buf_sents[i].str_sent=(wchar_t*)realloc(text_p->buf_sents[i].s
tr_sent,(text_p->buf_sents[i].leng_str+1)*sizeof(wchar_t));
                    for(k=text_p->buf_sents[i].leng_str;k>j;k--){

text_p->buf_sents[i].str_sent[k+1]=text_p->buf_sents[i].str_sent[k];
                    }
                    text_p->buf_sents[i].str_sent[j]='1';
                    if(*(is_num)=='2')
```

```
                        text_p->buf_sents[i].str_sent[j+1]='0';
                    else
                        text_p->buf_sents[i].str_sent[j+1]='1';
                }

if((*(is_num)=='4')||(*(is_num)=='5')||(*(is_num)=='6')||(*(is_num)=='
7')) {
                    text_p->buf_sents[i].str_sent = (wchar_t *)
realloc(text_p->buf_sents[i].str_sent,

(text_p->buf_sents[i].leng_str + 2) *

sizeof(wchar_t));
                    for (k = text_p->buf_sents[i].leng_str; k > j; k-
-) {
                        text_p->buf_sents[i].str_sent[k + 2] =
text_p->buf_sents[i].str_sent[k];
                    }
                    text_p->buf_sents[i].str_sent[j] = '1';
                    switch (*(is_num)) {
                        case L'4':
                            text_p->buf_sents[i].str_sent[j + 1] =
'0';
                            text_p->buf_sents[i].str_sent[j + 2] =
'0';
                            break;
                        case L'5':
                            text_p->buf_sents[i].str_sent[j + 1] =
'0';
                            text_p->buf_sents[i].str_sent[j + 2] =
'1';
                            break;
                        case L'6':
                            text_p->buf_sents[i].str_sent[j + 1] =
'1';
                            text_p->buf_sents[i].str_sent[j + 2] =
'0';
                            break;
```

```
                        case L'7':
                            text_p->buf_sents[i].str_sent[j + 1] =
'1';
                            text_p->buf_sents[i].str_sent[j + 2] =
'1';
                            break;
                    }
                }
                if((*(is_num)=='8')||(*(is_num)=='9')){

text_p->buf_sents[i].str_sent=(wchar_t*)realloc(text_p->buf_sents[i].s
tr_sent,(text_p->buf_sents[i].leng_str+3)*sizeof(wchar_t));
                    for(k=text_p->buf_sents[i].leng_str;k>j;k--){

text_p->buf_sents[i].str_sent[k+3]=text_p->buf_sents[i].str_sent[k];
                    }
                    text_p->buf_sents[i].str_sent[j]='1';
                    text_p->buf_sents[i].str_sent[j+1]='0';
                    text_p->buf_sents[i].str_sent[j+2]='0';
                    if(*(is_num)=='8')
                        text_p->buf_sents[i].str_sent[j+3]='0';
                    else
                        text_p->buf_sents[i].str_sent[j+3]='1';


                }
            }
        }
    }
}
```

Файл: task_3.h

```
#ifndef UNTITLED_TASK_3_H
#define UNTITLED_TASK_3_H

void Task_3(Text* text_p);
#endif
```

Файл: task_4.c

```
#include <stdlib.h>
#include <wchar.h>
#include <locale.h>
#include "structs.h"
void Task_4(Text* text_p){
    int i=0,j,k,flag;
    while(i<text_p->leng_text){ //перебираем предложения
        flag=1,j=0;
        while(flag&&j<text_p->buf_sents[i].leng_str) {//перебираем
символы внутри .str_sent
            wchar_t c = text_p->buf_sents[i].str_sent[j];
            if(c=='1'||c=='3'||c=='5'||c=='7'||c=='9'){
                for(k=i;k<text_p->leng_text-1;k++){
                    text_p->buf_sents[k]=text_p->buf_sents[k+1];
                }
                text_p->leng_text-=1;
                flag=0;
            }
            else {
                j++;
            }
        }
        if(flag==1)i++;
    }
}
```

Файл: task_4.h

```
#ifndef UNTITLED_TASK_4_H
#define UNTITLED_TASK_4_H

void Task_4(Text* text_p);
#endif //UNTITLED_TASK_4_H
```

Файл: Makefile

```
all: input_f.o task_1.o task_2.o task_3.o task_4.o menu.o
	gcc input_f.c task_1.c task_2.c task_3.c task_4.c menu.c -o
menu
input_f.o: input_f.c input_f.h
	gcc -c input_f.c
task_1.o: task_1.c task_1.h
	gcc -c task_1.c
task_2.o: task_2.c task_2.h
	gcc -c task_2.c
task_3.o: task_3.c task_3.h
	gcc -c task_3.c
task_4.o: task_4.c task_4.h
	gcc -c task_4.c
menu.o: menu.c input_f.h task_1.h task_2.h task_3.h task_4.h
	gcc -c menu.c
clean:
	rm *.o menu
```