**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 _____ Ильин Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

Студент Ильин Денис

Группа 0382

Тема работы: обработка строк на языке Си

Исходные данные:

Вариант 6

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. Распечатать каждое слово которое встречается не более одного раза в тексте.
2. Каждую подстроку в тексте имеющую вид “<день> <месяц> <год> г.” заменить на подстроку вида “ДД/ММ/ГГГГ”. Например, подстрока “20 апреля 1889 г.” должна быть заменена на “20/04/1889”.
3. Отсортировать предложения по произведению длин слов в предложении.

4. Удалить все предложения, которые содержат символ ‘#’ или ‘№’, но не содержат ни одной 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.

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: 24.12.2020

Дата защиты реферата: 26.12.2020

Студент _____ Ильин Д.А.

Преподаватель _____ Жангиров Т.Р.

# АННОТАЦИЯ

В процессе выполнения курсовой работы была создана программа для обработки введённого пользователем текста на языке Си. Для хранения текста, предложений, слов использовались структуры. Обработка и вывод текста производились при помощи использования функций следующих стандартных библиотек: *stdlib.h, wchar.h, locale.h*. Для удобства пользователя реализован вывод на консоль контекстного меню выбора необходимой команды обработки текста. При некорректном вводе выводится соответствующее сообщение. Для хранения текста память выделяется динамически. Также был написан Makefile для удобной сборки программы.

**СОДЕРЖАНИЕ**

# ВВЕДЕНИЕ

Данная программа получает на вход через консоль некоторый текст, после чего делает первоначальную его обработку (удаляются повторяющиеся предложения). После чего предлагает пользователю ввести одну из 6ти имеющихся команд, а именно:

1 — Распечатать каждое слово которое встречается не более одного раза в тексте.

2 — Каждую подстроку в тексте имеющую вид “<день> <месяц> <год> г.” заменить на подстроку вида “ДД/ММ/ГГГГ”. Например, под строка “20 апреля 1889 г.” должна быть заменена на “20/04/1889”.

3 — Отсортировать предложения по произведению длин слов в предложе нии.

4 — Удалить все предложения, которые содержат символ ‘#’ или ‘№’, но не содержат ни одной цифры.

0 — Вывести получившийся текст.

9 — Закончить выполнение программы

Для каждой команды была реализована соответствующая функция. Также была реализована функция считывания текста и удаления предложений по индексу.

## ХОД ВЫПОЛНЕНИЯ РАБОТЫ.

### Структуры для считывания текста

В программе были созданы структуры *Words*, *Sentence* и *Text,* которые впоследствии были переименованы, при помощи оператора typedef в *WORD, SENT* и *TEXT* соответственно.

Поля *WORD*:

- *wchar_t * word* — динамический массив символов.
- *int len* — размер динамического массива символов.

Поля *SENT*:

- *WORD * sent* — динамический массив слов.
- *int spe* — количество специальных символов(№ или #).
- *int in* – количество чисел встречающихся в предложении.
- *int len* — размер динамического массива слов.
- *long long int mass* — произдедение длин слов в предложении.

Поля *TEXT*:

- *SENT * text* — динамический массив предложений.
- *int len* — количество предложений.

### Ввод текста.

Первым делом устанавливается русская локализация, дабы была возможна обработка кириллицы. После этого пользователю предлагается ввести текст и прописываются символы конца текста- это точка с переносом строки идущие подряд.

Для считывания предложений была реализована функция *read(),* которая в цикле считывает посимвольно введённые слова, из которых формируются слова(элементы структуры *WORD*), из которых формируются предложения. Данная функция считывает только до точки, также если встречаются цифры или специальные символы, то внутри структуры *SENT* увеличиваются

соответствующие параметры(*spe* — отвечает за специальные символы, *in* — отвечает за числа).

Изначальную инициализацию для соответствующих структур проводят функции: *initializeWord(), initializeSent(), initializeText()*, там присваиваются изначальные значения соответствующих переменных, а также выделяется память с помощью метода *malloc*, первоначального размера 100. Далее при необходимости увеличения памяти используется метод *realloc*.

**Первоначальная обработка текста.**

Реализована функция *create_text()*, которая в цикле, при помощи функции *read()*, формирует текст. Далее программа удаляет одинаковые предложения, изначально сверяя их друг с другом при помощи функции *wcscasecmp()*, сверяющей строки без учёта регистра. Для удаления предложений была реализована функция *shift()*, которая помимо удаления освобождает память, а также сдвигает предложения, дабы не было проблемных мест.

**Вывод полученного текста**

Для этого была реализована специальная функция *print(),*которая в цикле выводит пословно текст, который ей передаётся.

**Функция *print_word()***

Данная функция в цикле проходится по тексту и для каждого слова считает сколько раз оно в тексте встречается, если оно встречается только один раз, то функция его выводит. Проверка слов происходит при помощи функции *wcscasecmp()*, которая сравнивает слова без учёта регистра.

**Функция *fix_data()***

Данная функция ищет в тексте некоторую подстроку, формата “<день> <месяц> <год> г.” и меняет на подстроку вида “ДД/ММ/ГГГГ”. Например, подстрока “20 апреля 1889 г.” должна быть заменена на “20/04/1889”. Для этого используется проверка каждого слова по отдельности, на правильность его формата(слова разделены пробелами в изначальном тексте).

- Первое слово из данной подстроки должно быть длинны 2 и содержать в себе только числа .
- Второе слово должно быть из списка слов месяцов, в котором находятся все месяцы прописанные на кириллице(сравнение происходит без учёта регистра).
- Третье слово должно быть длинны 4 и содержать в себе только числа.
- Последнее слово должно быть буквой г.

После чего пробелы заменяются на „/“, месяц заменяется на его индекс в списке месяцов,  буква г удаляется, а цифры остаются на месте.

**Функция *del_spe()***

Данная функция удаляет предложения в которых имеется специальный символ, но без чисел. Функция в цикле проверяет есть ли специальный символ и есть ли числа, если попадается предложение, которое нужно удалить, то удаляет его при помощи функции *shift()*.

**Функция *text_sort()***

Данная функция сортирует предложения по возрастанию их параметра *mass.*

**Функция *main()***

В функции ставится русская локализация, после чего отправляется запрос пользователю на текст. Далее выводится в консоль описание команд и и просьба ввести некоторые из них(одной из команд является команда окончания работы программы). После чего при помощи оператора выбора *switch* вызыва-

ется одна из функций описаных выше. При неправильно введённой команде программа выводит соответствующее сообщение.

## ЗАКЛЮЧЕНИЕ

Была разработана программа, которая формирует текст из того, что ввёл пользователь, запрашивает команды, по которым обрабатывает его некоторым образом. Таким образом цель лабораторной работы была достигнута.

В результате данной курсовой работы были изучены основные принципы обработки строк на языке Си. Получены навыки использования структур, оператора выбора *switch*, создания *Makefile*, работы с динамической памятью и работы с функциями заголовочных файлов *wchar.h* и *locale.h*.

## 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.

## ПРИЛОЖЕНИЕ А

## ПРИМЕР РАБОТЫ ПРОГРАММЫ

Вывод с консоли после вызов утилиты make:

```
denis@Denis-GP62-2QE:~/C/KURS$ make clean
rm -rf *.o kurs
denis@Denis-GP62-2QE:~/C/KURS$ make
gcc -c main.c
gcc -c functions.c
gcc -c struct.c
gcc -c text_maker.c
gcc main.o functions.o struct.o text_maker.o -o kurs
denis@Denis-GP62-2QE:~/C/KURS$
```

Вывод меню после запуска исполняемого файла kurs:

```
denis@Denis-GP62-2QE:~/C/KURS$ ./kurs
Здравствуйте!
Введите текст, с которым нужно работать(текст должен оканчиваться на точку и символ пе
реноса строки идущими подряд).
```

После ввода текста.

```
denis@Denis-GP62-2QE:~/C/KURS$ ./kurs
Здравствуйте!
Введите текст, с которым нужно работать(текст должен оканчиваться на точку и символ перен
оса строки идущими подряд).
07 ДЕКАБРЯ 1889 г. -- Ну Ну что, князь, Генуя и Лукка стали не больше как поместья, помес
тья фамилии Буонапарте.
Введите последовательность команд обработки текста из предложеных, а затем нажмите ENTER:
1- Распечатать каждое слово которое встречается не более одного раза в тексте.
2- Каждую подстроку в тексте имеющую вид “<день> <месяц> <год> г.” заменить на подстроку
вида “ДД/ММ/ГГГГ”. Например, подстрока “20 апреля 1889 г.” должна быть заменена на “20/04
/1889”.
3- Отсортировать предложения по произведению длин слов в предложении.
4- Удалить все предложения, которые содержат символ ‘#’ или ‘№’, но не содержат ни одной
цифры.
0- Напечатать получившийся текст.
9- Закончить работу программы.
```

Пример выполнения всех опций и обработки неправильно введённой команды.

```
denis@Denis-GP62-2QE:~/C/KURS$ ./kurs
Здравствуйте!
Введите текст, с которым нужно работать(текст должен оканчиваться на точку и символ перен
оса строки идущими подряд).
07 ДЕКАБРЯ 1889 г. -- Ну Ну что, князь, Генуя и Лукка стали не больше как поместья, помес
тья фамилии Буонапарте.
Введите последовательность команд обработки текста из предложеных, а затем нажмите ENTER:
1- Распечатать каждое слово которое встречается не более одного раза в тексте.
2- Каждую подстроку в тексте имеющую вид “<день> <месяц> <год> г.” заменить на подстроку
вида “ДД/ММ/ГГГГ”. Например, подстрока “20 апреля 1889 г.” должна быть заменена на “20/04
/1889”.
3- Отсортировать предложения по произведению длин слов в предложении.
4- Удалить все предложения, которые содержат символ ‘#’ или ‘№’, но не содержат ни одной
цифры.
0- Напечатать получившийся текст.
9- Закончить работу программы.
2
Введите следующий набор команд.
0
07/12/1889. -- Ну Ну что, князь, Генуя и Лукка стали не больше как поместья, поместья фам
илии Буонапарте.
Введите следующий набор команд.
```

```
9
denis@Denis-GP62-2QE:~/C/KURS$ ./kurs
Здравствуйте!
Введите текст, с которым нужно работать(текст должен оканчиваться на точку и символ перен
оса строки идущими подряд).
1233.1233. 1 1 1. 22 22 22. 134#. №dff.
Введите последовательность команд обработки текста из предложеных, а затем нажмите ENTER:
1- Распечатать каждое слово которое встречается не более одного раза в тексте.
2- Каждую подстроку в тексте имеющую вид “<день> <месяц> <год> г.” заменить на подстроку
вида “ДД/ММ/ГГГГ”. Например, подстрока “20 апреля 1889 г.” должна быть заменена на “20/04
/1889”.
3- Отсортировать предложения по произведению длин слов в предложении.
4- Удалить все предложения, которые содержат символ ‘#’ или ‘№’, но не содержат ни одной
цифры.
0- Напечатать получившийся текст.
9- Закончить работу программы.
0
1233. 1 1 1. 22 22 22. 134#. №dff.
Введите следующий набор команд.
30
 1 1 1.1233. 134#. №dff. 22 22 22.
Введите следующий набор команд.
```

```
3- Отсортировать предложения по произведению длин слов в предложении.
4- Удалить все предложения, которые содержат символ ‘#’ или ‘№’, но не содержат ни одной
цифры.
0- Напечатать получившийся текст.
9- Закончить работу программы.
0
1233. 1 1 1. 22 22 22. 134#. №dff.
Введите следующий набор команд.
30
 1 1 1.1233. 134#. №dff. 22 22 22.
Введите следующий набор команд.
40
 1 1 1.1233. 134#. 22 22 22.
Введите следующий набор команд.
1
1233 134#
Введите следующий набор команд.
t
Введённая команда "t" некорректна.
Для завершения работы программы введите 9.
Введите следующий набор команд.
9
denis@Denis-GP62-2QE:~/C/KURS$
```

# ПРИЛОЖЕНИЕ Б КОД ПРОГРАММЫ

Файл main.c:

```
#include  "struct.h"
#include "TextMaker.h"
#include "functions.h"

int main() {
    setlocale(LC_ALL, "ru_RU.UTF-8");
    wprintf(L"Здравствуйте!\nВведите текст, с которым нужно работать(текст должен оканчиваться на точку и символ переноса строки идущими подряд).\n");
    TEXT work_text = create_text();
    wprintf(L"Введите последовательность команд обработки текста из предложеных, а затем нажмите ENTER:\n1- Распечатать каждое слово которое встречается не более одного раза в тексте."
            "\n2- Каждую подстроку в тексте имеющую вид “<день> <месяц> <год> г.” заменить на подстроку вида “ДД/ММ/ГГГГ”. Например, подстрока “20 апреля 1889 г.” должна быть заменена на “20/04/1889”."
            "\n3- Отсортировать предложения по произведению длин слов в предложении."
            "\n4- Удалить все предложения, которые содержат символ ‘#’ или ‘№’, но не содержат ни одной цифры."
            "\n0- Напечатать получившийся текст."
            "\n9- Закончить работу программы.\n");
    wchar_t command = getwchar();
    while (command != L'9') {
        switch (command) {
            case L'0':
                print(&work_text);
                break;
            case L'1':
                print_word(&work_text);
                break;
            case L'2':
                fix_data(&work_text);
                break;
            case L'3':
                text_sort(&work_text);
                break;
            case L'4':
                del_spe(&work_text);
                break;
            case '\n':
                wprintf(L"Введите следующий набор команд.\n");
                break;
            default:
                wprintf(L"Введённая команда \"%lc\" некорректна.\nДля завершения работы программы введите 9.\n", command);
                break;
        }
        command = getwchar();
    }
    return 0;
}
```

Файл TextMaker.c:

```
#include "TextMaker.h"

SENT read(){
    WORD inp_word;
    SENT inp_sent;
    int size = 100; //размер выделенной памяти для inp_word и inp_sent
    wchar_t inp;

    initializeSent(&inp_sent);
    initializeWord(&inp_word);
    inp = getwchar();
    if (inp == L'\n'){
        inp_word.word[inp_word.len++] = inp;
        inp_word.word[inp_word.len++] = L'\0';
        inp_sent.sent[0] = inp_word;
        return inp_sent;
    }
    while (inp != L'.'){
        if ((inp_word.len+5> size) || (inp_sent.len+5> size)) {
            size += 1000;
            inp_word.word = realloc(inp_word.word, size * sizeof(wchar_t));
            inp_sent.sent = realloc(inp_sent.sent, size * sizeof(WORD));
        }
        if (inp<100) {
            if (isDigit(inp)) {
                inp_sent.in++;
                //wprintf(L"%ls %d\n", inp_sent.sent[0].word, inp_sent.in);
            }
        }
        if (( inp == L'#')||( inp == L'№')){
            inp_sent.spe++;
            //wprintf(L"%ls %d\n", inp_sent.sent[0].word, inp_sent.spe);
        }
        if ((inp == L' ') || (inp == L',')){
            inp_word.word[inp_word.len++] = L'\0';
            if (inp_sent.mass < 100000000000000) {
                if (inp_word.len > 1) {
                    inp_sent.mass *= inp_word.len - 1;
                }
            }
            //wprintf(L"%ld\n", inp_sent.mass);
            inp_sent.sent[inp_sent.len++] = inp_word;
            initializeWord(&inp_word);
            inp_word.word[inp_word.len++] = inp;
            inp_word.word[inp_word.len++] = L'\0';
            inp_sent.sent[inp_sent.len++] = inp_word;
            initializeWord(&inp_word);
            inp = getwchar();
        }
        else{
            inp_word.word[inp_word.len++] = inp;
            //inp_word.word[inp_word.len++] = L'\0';
            inp = getwchar();
        }
        //wchar_t * arr = inp_word.word;
        //wprintf(L"%ls %c %c\n", arr, inp_word.word[inp_word.len - 1], inp);
    }
    inp_word.word[inp_word.len++] = L'\0';
    if (inp_sent.mass < 100000000000000) {
        if (inp_word.len > 1) {
```

```
            inp_sent.mass *= inp_word.len - 1;
        }
    }
    //wprintf(L"%lld w\n", inp_sent.mass);
    inp_sent.sent[inp_sent.len++] = inp_word;
    return inp_sent;
}

void shift(TEXT * lst, int i){
    for (int j = 0; j < lst->text[i].len; ++j) {
        free(lst->text[i].sent[j].word);
    }
    for (;i <lst->len;i++){
        lst->text[i] = lst->text[i+1];
    }
    int t = (lst->len)-1;
    free(lst->text[t].sent);
    lst->len--;
}

TEXT create_text(){
    TEXT inp_text;
    initializeText(&inp_text);
    SENT inp_sent;
    int size = 100;
    do {
        if (inp_text.len + 5 > size){
            size *= 2;
            inp_text.text = realloc(inp_text.text, size * sizeof(SENT));
        }
        inp_sent = read();
        inp_text.text[inp_text.len++] = inp_sent;
        //wprintf(L"%d\n", inp_text.len);
        for (int i = 0; i < inp_sent.len; i++){
            //wprintf(L"%ls %d\n", inp_sent.sent[i], inp_text.len);
        }
    }while (inp_sent.sent[0].word[0] != L'\n');

    for (int i = 0; i < inp_text.len - 2; i++){
        for (int j = i+1; j < inp_text.len - 1;){
            //wprintf(L"%d %d\n", inp_text.text[i].len, inp_text.text[j].len);
            if (inp_text.text[i].len == inp_text.text[j].len){
                int help = 0;
                for (int k = 0; k < inp_text.text[i].len; k++){
                    if
(wcscasecmp(inp_text.text[i].sent[k].word,inp_text.text[j].sent[k].word)){
                        help++;
                    }
                }
                if (help == 0){
                    shift(&inp_text, j);
                } else j++;
            }
            else{
                j++;
            }
        }
    }
    return inp_text;
}
```

Файл TextMaker.h:

```
#include  <wchar.h>
#include <locale.h>
#include <stdlib.h>
#include "struct.h"

void shift(TEXT * lst, int i);
TEXT create_text();
```

Файл struct.c:

```
#include  "struct.h"

void initializeWord(WORD * lst){
   lst->word = malloc(100 *
sizeof(wchar_t));
   lst->len = 0;
}

void initializeSent(SENT * lst){
   lst->sent = malloc(100 *
sizeof(WORD));
   lst->len = 0;
   lst->in = 0;
   lst->spe = 0;
   lst->mass = 1;
}

void initializeText(TEXT * lst){
   lst->text = malloc(100 *
sizeof(SENT));
   lst->len = 0;
}

int isDigit(char qwes){
   if ((qwes >= '0') && (qwes <=
'9')) {
       return 1;
   } else return 0;
}
```

Файл struct.h:

```
#include  <wchar.h>
#include <locale.h>
#include <stdlib.h>
#ifndef END_WERS_KURS_J_H
#define END_WERS_KURS_J_H
struct Words {
    wchar_t * word;
    int len; // длина слова
};

typedef struct Words WORD;

struct Sentence {
    WORD * sent;
    int spe; // есть ли специальныеслимволы в предложении
    int in; // есть ли числа в предложении
    int len; // количество слов в предложении
    long long int mass;
};

typedef struct Sentence SENT;

struct Text {
    SENT * text;
    int len; // количество предложений в тексте
};

#endif

typedef struct Text TEXT;
void initializeWord(WORD * lst);
void initializeSent(SENT * lst);
void initializeText(TEXT * lst);
int isDigit(char qwes);
```

Файл functions.c:

```
#include "functions.h"

void print(TEXT * work_text){
    int i = 0;
    if (work_text->text[i].sent[0].word[0] != L'\n') {
        do {
            for (int j = 0; j < work_text->text[i].len; j++) {
                wchar_t *arr = work_text->text[i].sent[j].word;
                wprintf(L"%ls", arr);
            }
            i++;
            wprintf(L".");
        } while (work_text->text[i].sent[0].word[0] != L'\n');
    }
    wprintf(L"\n");
}

void print_word(TEXT * work_text){
    for (int i = 0; i < work_text->len-1; ++i) {
        for (int j = 0; j < work_text->text[i].len; ++j) {
            int flag = 0;
            if ((work_text->text[i].sent[j].word[0] != L' ') && (work_text->text[i].sent[j].word[0] !=
L',')) {
                for (int k = 0; k < work_text->len - 1; ++k) {
                    for (int l = 0; l < work_text->text[k].len; ++l) {
                        if (!(wcscasecmp(work_text->text[i].sent[j].word, work_text-
>text[k].sent[l].word))){
                            flag++;
                        }
                    }
                }
                if (flag == 1){
                    wprintf(L"%ls ", work_text->text[i].sent[j].word);
                }
            }
        }
    }
    wprintf(L"\n");
}

void fix_data(TEXT * work_text){
    wchar_t month[12][30] = {L"Января", L"Февраля", L"Марта", L"апреля", L"Мая", L"Июня",
L"Июля", L"Августа", L"Сентября", L"Октября", L"Ноября", L"Декабря"};
    for (int i = 0; i < work_text->len-1; ++i) {
        for (int j = 0; j < work_text->text[i].len - 6; ++j) {
            if (work_text->text[i].sent[j].len == 3 && isDigit(work_text->text[i].sent[j].word[0]) &&
isDigit(work_text->text[i].sent[j].word[1])){
                if (work_text->text[i].sent[j+4].len == 5 && isDigit(work_text-
>text[i].sent[j+4].word[0]) && isDigit(work_text->text[i].sent[j+4].word[1]) &&
isDigit(work_text->text[i].sent[j+4].word[2]) && isDigit(work_text->text[i].sent[j+4].word[3])){
                    if (work_text->text[i].sent[j+6].word[0] == L'г'){
                        int nam;
                        for (nam = 0; nam < 12; ++nam) {
                            if (!wcscasecmp(month[nam], work_text->text[i].sent[j+2].word)) {
                                work_text->text[i].sent[j + 1].word = L"/\0";
```

```
                        work_text->text[i].sent[j + 3].word = L"/\0";
                        work_text->text[i].sent[j + 5].word = L"\0";
                        if (nam<9) {
                            work_text->text[i].sent[j + 2].word[0] = '0';
                            work_text->text[i].sent[j + 2].word[1] = (nam+1) +'0';
                            work_text->text[i].sent[j + 2].word[2] = '\0';
                        }
                        else {
                            work_text->text[i].sent[j + 2].word[0] = L'1';
                            work_text->text[i].sent[j + 2].word[1] = ((nam+1)%10) +'0';
                            work_text->text[i].sent[j + 2].word[2] = '\0';
                        }
                        free(work_text->text[i].sent[j+6].word);
                        free(work_text->text[i].sent[j+7].word);
                        work_text->text[i].len = work_text->text[i].len - 2;
                    }
                }
            }
        }
    }
}

void del_spe(TEXT * work_text){
    for (int i = 0; i < work_text->len;) {
        //wprintf(L"%d %d\n", work_text->text[i].spe, work_text->text[i].in);
        if ((work_text->text[i].spe != 0) && (work_text->text[i].in == 0)){
            shift(work_text, i);
        }
        else i++;
    }
}

void text_sort(TEXT * work_text){
    for (int i = 0; i < work_text->len-1; ++i) {
        long long int minmass, index;
        index = i;
        minmass = work_text->text[i].mass;
        //wprintf(L"%d\n", minmass);
        for (int j = i+1; j < work_text->len-1; ++j) {
            if (minmass > work_text->text[j].mass){
                index = j;
                minmass = work_text->text[j].mass;
            }
        }
        SENT help;
        help = work_text->text[i];
        work_text->text[i] = work_text->text[index];
        work_text->text[index] = help;
    }
}
```

Файл functions.h:

```
#include "TextMaker.h"

void print(TEXT * work_text);
void print_word(TEXT * work_text);
void fix_data(TEXT * work_text);
void del_spe(TEXT * work_text);
void text_sort(TEXT * work_text);
```

Makefile:

```
kurs: main.o functions.o struct.o text_maker.o
	gcc main.o functions.o struct.o text_maker.o -o kurs

main.o: main.c struct.h
	gcc -c main.c

functions.o: functions.c functions.h
	gcc -c functions.c

struct.o: struct.c struct.h
	gcc -c struct.c

text_maker.o: text_maker.c text_maker.h
	gcc -c text_maker.c

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