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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка строк на языке Си

Студентка гр. 0382 _____ Здобнова К.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студентка Здобнова К.Д.

Группа 0382

Тема работы: Обработка строк на языке Си.

Исходные данные:

Программе на вход подается текст (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).

Содержание пояснительной записки:

Обязательны разделы «Содержание», «Введение», «Ход работы», «Заключение», «Тестирование»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 14.11.2020

Дата сдачи курсовой: 29.12.2020

Дата защиты реферата: 29.12.2020

Студентка _____ Здобнова К.Д.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

Курсовая работа представляет собой программу на языке Си для обработки текста. Текст считывается с консоли, пользователю на выбор предоставляются несколько команд для обработки считанного текста.

Для хранения текста реализованы структуры Text и Sentence, память выделяется динамически с помощью *malloc* и *realloc* в ходе считывания посимвольно из консоли.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью работы является написание корректной программы для работы с текстовыми данными и функций для их обработки. Так как программа должна работать с кириллицей, в коде использовался символьный тип *wchar_t* из библиотеки *wchar.h*. Для вывода на консоль предварительно устанавливалась *setlocale(LC_ALL, “”).* Память для хранения текста выделялась динамически с помощью *malloc* и *realloc*, по окончанию выполнения программы память освобождалась с помощью функции free из библиотеки *stdlib.h*.

Каждая подзадача программы выделена в отдельный файл. Сборка программы реализуется с помощью утилиты make.

1. ЗАДАНИЕ

Вариант 4.

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) В каждом предложении заменить первое слово на второе слово из предыдущего предложения. Для первого предложения, второе слово надо брать из последнего.
2) Отсортировать предложения по длине третьего слова. Если слов меньше трех, то длина третьего слова равняется нулю.
3) Вывести на экран все предложения, в которых в середине слова встречаются цифры. Данные слова нужно выделить зеленым цветом.
4) В каждом предложении, в слове, все символы, которые встречаются несколько раз подряд заменить одним таким символом.

Все сортировки и операции со строками должны осуществляться с использованием функций стандартной библиотеки. Использование

собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

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

2. ХОД РАБОТЫ

2.1. Функции main():

В стандартный поток вывода печатается сообщение «Введите текст». Пользователь вводит текст, придерживаясь правилам разделения слов и предложений: предложения разделены точкой, слова либо запятой, либо пробелом. Создается переменная *inp_text* типа *struct Text,* память для полей которой выделяется с помощью *malloc.* После чего для пользователя печатается сообщение следующего вида:

Выберете команду:

1: В каждом предложении заменить первое слово на второе слово из предыдущего.

2: Отсортировать предложения по длине третьего слова.

3: Вывести на экран текст, выделяя зеленым слова, в середине которых есть цифры.

4: Убрать в словах повторяющиеся символы.

5: Вывести текст на экран.

6: Выйти из редактора текста.

В цикле while обрабатываются запросы пользователя, пока не будет выбрана команда выхода из редактора текста.

2.2. Структуры

Структура *Text* – структура для хранения предложений и его размер. Поля:

sentences (тип *struct Sentence**) – динамический массив предложений и информации о них.

len_text (тип *int*) – число, равное количеству предложений в тексте.

Структура *Sentence* – структура для хранения предложения, имеющая следующие поля:

words (тип *wchar_t***) – динамический массив, хранящий указатели на слова предложения.

separator (тип *wchar_t**) – динамический массив, записывающий разделители текста.

num_of_words (тип *int*) – число, хранящие количество слов в предложении.

len_of_3 (тип *int*) – количество символов в третьем слове предложения.

2.3. Функции

Функция *get_text()*

Функция, предназначенная для считывания текста из стандартного потока вывода. Переменные *text_buffer, words_buffer, sep_buf* предназначены для отслеживания заполнения памяти для полей Sentence. Массив word (тип *wchar_t**) – хранит указатели на область памяти, куда записаны слова.

В переменную *inp_sym* (тип *wchar_t*) в цикле *while* считываются символы текста. Считывания происходит с помощью *getwchar()*. В *words_in_sent* (тип *int*) является счетчиком для количества слов в предложении, *letters_in_sent* (тип *int*) – для количества букв, *letters_in_word* (тип *int*) для количества букв в слове, а *num_of_sent* (тип *int*) считает порядок предложения для дальнейшей записи.

Пока не встречаются разделители предложения – записываем слова, затем записываем разделитель предложения, и так до тех пор пока не встречается символ «.». Дальше с помощью цикла for проверяется одно из условий – не повторения предложения: посимвольно сравниваются предыдущие строки (*prev*) с текущей, в случае неравенства в *inp_text.sentences* записывается новое предложение.

Память для динамический массивов выделяется с помощью *malloc* и *realloc*.

В конце работы функция определяет поле *len_text* и возвращает считанный текст.

Функция print_text()

Функция предназначена для вывода текста на экран. Принимает аргументом *struct Text inp_text*. Так как предложения хранятся с помощью

полей words и separator, они поочередно печатаются (так как после каждого предложения обязательно следует один из разделителей текста).

Функция replace()

Принимает аргументом структуру *struct Text text*. Предназначена для того, чтобы в каждом предложении заменить первое слово на второе слово из предыдущего предложения. Для первого предложения второе слово берется из последнего. С помощью цикла forпроверяет, состоит ли предложение из нескольких или более слов. Если да, то последующему слову заменяет его первое – на второе. Если нет, то продолжает цикл for со следующего номера. Чтобы не выходить за границы массива, первое предложение обрабатывается отдельно.

Функция ничего не возвращает.

Функция *q_sort()*

Принимает аргументом структуру *struct Text text*. Предназначена для сортировки текста по длине третьего предложения(если в предложении нет третьего слова, то поле *len_of_3* равняется нулю) Вызывает функцию *compare*.

Функция *compare*

Сравнивает длину двух чисел. Возвращает -1, 0 или 1 в зависимости от значения чисел.

Функция *set_color()*

Принимает аргументом структуру *struct Text text*. Предназначена для того, чтобы вывести на экран предложения, выделяя зеленым слов, в середине которых встречаются цифры. Проверка происходит посимвольно у поля words с помощью функции стандартной библиотеки *isdigit()*. Помеченное слово выделяет зеленным при выводе (\033[42m). Переменная *flag* (тип *int*) служит для обозначения того, что в слове встретилась цифра.

Функция *double_letters()*

Принимает аргументом структуру *struct Text text*. Предназначена для того, чтобы в словах удалять повторяющиеся друг за другом символы.

Переменные *left* и *right* служат в роли указателей на символы в строке: пока переменная *left* не станет равной длине слова, буде происходит обработка слова. Если встречаются повторяющиеся символы, то лишние заменяются на символ – первый отличный справа (в цикле *while* (*k* <= *same*)).

Так как в этой функции может изменится длина слова, то в конце обработки каждого предложения обновляется поле *len_of_3*.

Линковка

Для удобства работы программа была разбита на файлы:

Заголовочные файлы:

double_letters.h

q_sort.h

replace.h

set_color.h

text_func.h

Файлы реализации:

double_letters.c

q_sort.c

replace.c

set_color.c

text_func.c

Сборка программы осуществлялась с помощью *Makefile*.

Код программы представлен в приложении А.

2.4 Тестирование

В табл. 1 представлены результаты тестирования.

табл.1

№т	Входные данные	Выходные данные	Комментарий
1	Caaats is a 2019 musical film.Cats.Cats,was theatttrically,releeased in	was.Cats is a 2019 musical film.is,a young white cat,is drop5ed in	Текст сначала был отсортирован, затем в словах были

	the United Kingdom.Cats.Victoria,a young white cat,is dropp55ed in the streets. 2 4 1 3 6	the strets.a,was theatricaly,released in the United Kingdom.	убраны повторяющиеся символы. В каждом предложении заменили первое слово на второе слово из предыдущего. В последствии текст был выведен с выделением слов с цифрами в середине.
2	Раз.Раз.Раз два.Раз два три.Раз два три четыре.Раз два три четыре пять. 5 1 5 6	Раз.Раз два.Раз два три.Раз два три четыре.Раз два три четыре пять. два.Раз два.два два три.два два три четыре.два два три четыре пять.	Первой командой текст вывели на экран. Далее поменяли первые и вторые слова согласно условию. Затем текст снова напечатали.
3	Ddf22d,qa erddddf.Fjd ll,ff.Re.Ttd12gd,dfdfd.Fjd ll,ff.Fjd ll ff.Ttdgd,dfdfd rtt33t re.4rer Fdf errwffdf. 4 2 3 6	Ttd12gd,dfdfd.Re.Fjd l f.Fjd l,f.Ddf2d,qa erdf.Ttdgd,dfdfd rt3t re.4rer Fdf erwfdf.	Программа убрала повт. символы, отсортировала предложения по длине 3-го слова. Вывела на экран текст с зеленым выделением слов.

ЗАКЛЮЧЕНИЕ

Были изучены структуры языка Си, с помощью которых можно сохранять данные в памяти. Реализован код для работы с текстовыми данными, включая кириллицу. Код разделен на подзадачи, сборка которого осуществляется с помощью утилиты make.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

1. файл main.c

```
#include <stdio.h>
#include <wchar.h>
#include <wctype.h>
#include <locale.h>
#include <stdlib.h>
#include "structs.h"
#include "text_func.h"
#include "double_letters.h"
#include "q_sort.h"
#include "replace.h"
#include "set_color.h"
#include "text_func.h"

int main() {
    setlocale(LC_ALL, "");
    wchar_t command = L'0', space;
    struct Text inp_text;
    wprintf(L"Введите текст\n");
    inp_text = get_text();
    wprintf(L"Выберете команду:\n"
            "1: В каждом предложении заменить первое слово на второе
слово из предыдущего.\n"
            "2: Отсортировать предложения по длине третьего слова.\n"
            "3: Вывести на экран текст, выделяя зеленым слова, в середине
которых есть цифры.\n"
            "4: Убрать в словах повторяющиеся символы.\n"
            "5: Вывести текст на экран.\n"
            "6: Выйти из редактора текста.\n");
    while (command != L'6'){
        command = getwchar();
        space = getwchar();
        switch(command){
            case L'1': {
                replace(inp_text);
                break;
            }
            case L'2': {
                q_sort(inp_text);
                break;
            }
            case L'3': {
                set_color(inp_text);
                break;
            }
            case L'4': {
                double_letters(inp_text);
                break;
            }
            case L'5': {
                print_text(inp_text);
                break;
            }
```

```
            case L'6': {
                for (int i = 0; i < inp_text.len_text; i++){
                    free(inp_text.sentences[i].words);
                    free(inp_text.sentences[i].separator);
                }
                free(inp_text.sentences);
                break;
            }
            default:{
                wprintf(L"Данной опции не существует\n");
                break;
            }
        }
    }
    return 0;
}
```

2. файл double_letters.c

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

void double_letters(struct Text text){
    for (int i = 0; i < text.len_text; i++){ //в тексте
        for (int j = 0; j < text.sentences[i].num_of_words; j++){ //в
предложении
            int left = 0, right = 1, same = 0;
            while (left < wcslen(text.sentences[i].words[j])){
                if (text.sentences[i].words[j][left] ==
text.sentences[i].words[j][right]) {
                    right++;
                    same++;
                }
                else {
                    int k = 1;
                    while (k <= same){
                        text.sentences[i].words[j][left + k] =
text.sentences[i].words[j][right];
                        k++;
                    }
                    same = 0;
                    left++;
                    right = left + 1;
                }
            }
            if (j == 2){
                text.sentences[i].len_of_3 =
wcslen(text.sentences[i].words[j]);
            }
        }
    }
}
```

3. файл double_letters.h

```
void double_letters(struct Text text);
```

4. файл q_sort.c

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

int compare(const void * a, const void * b){
    struct Sentence *first = (struct Sentence*) a;
    struct Sentence *second = (struct Sentence*) b;
    return ((int)first->len_of_3 - (int)second->len_of_3);
}

void q_sort(struct Text text){
    qsort(text.sentences, text.len_text, sizeof(struct  Sentence),
compare);
}
```

5. файл q_sort.h

```
void q_sort(struct Text text);
int compare(const void * a, const void * b);
```

6. файл replace.c

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

void replace(struct Text text){
    for (int i = 0; i < text.len_text - 1; i++) { //в тексте
        if (text.sentences[i].num_of_words == 1)
            continue;
        else
            text.sentences[i + 1].words[0] = text.sentences[i].words[1];
    }
    if (text.sentences[text.len_text - 1].num_of_words > 1){
        text.sentences[0].words[0] = text.sentences[text.len_text -
1].words[1];

    }
}
```

7. файл replace.h

```
void replace(struct Text text);
```

8. файл set_color.c

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

```
#include <stdlib.h>
#include "structs.h"
#include "set_color.h"

#define NONE "\033[0m"
#define GREEN "\033[42m"


void set_color(struct Text text){
    int flag = 0;
    for (int i = 0; i < text.len_text; i++) { //в тексте
        for (int j = 0; j < text.sentences[i].num_of_words; j++) { //в
предложении
            if (wcslen(text.sentences[i].words[j]) != 1)
                for (int k = 1; k < wcslen(text.sentences[i].words[j]) -
1; k++){
                    if (iswdigit(text.sentences[i].words[j][k])){
                        wprintf(L"%s%ls", GREEN,
text.sentences[i].words[j]);
                        wprintf(L"%s%c", NONE,
text.sentences[i].separator[j]);
                        flag = 1;
                        break;
                    }
                }
            else{
                if (iswdigit(text.sentences[i].words[j][0])) {
                    wprintf(L"%s%ls", GREEN, text.sentences[i].words[j]);
                    wprintf(L"%s%c", NONE,
text.sentences[i].separator[j]);
                    flag = 1;
                }
            }
            if (flag == 0)
                wprintf(L"%ls%c", text.sentences[i].words[j],
text.sentences[i].separator[j]);
            flag = 0;
        }
    }
    wprintf(L"\n");
}
```

9. файл set_color.h

```
void set_color(struct Text text);
```

10. файл structs.h

```
struct Sentence{
    wchar_t **words;
    wchar_t *separator;
    int num_of_words;// количество слов в предложении
    int len_of_3; //длина третьего слова
};

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

11. файл text_func.c

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

#define BUFFER 100

struct Text get_text(){
    struct Text inp_text;
    int text_buffer = BUFFER, words_buffer = BUFFER, sep_buf = BUFFER;
    int words_in_sent = 0, letters_in_sent = 0, num_of_sent = 0,
letters_in_word = 0;
    wchar_t *word; //собираются слова
    inp_text.sentences  = malloc(text_buffer * sizeof(struct Sentence));
    wchar_t inp_sym;
    int point = 1, flag2 = 1, same = 0;
    while(point == 1){
        inp_sym = getwchar();
        if (inp_sym == L'\n') {
            point = 0;
            break;
        }
        struct Sentence sentence;
        sentence.words = malloc(words_buffer * sizeof(wchar_t*));
        sentence.separator = malloc(sep_buf * sizeof(wchar_t));
        sentence.num_of_words = 0;
        sentence.len_of_3 = 0;
        while(inp_sym != L'.') {
            word = malloc(words_buffer * sizeof(wchar_t));
            while (inp_sym != L' ' && inp_sym != L',' && inp_sym != L.')
{ //обработка слова
                word[letters_in_word] = inp_sym;
                letters_in_word++;
                if (letters_in_word + 5 == words_buffer){ //память для
счит. слова
                    word = realloc(word, words_buffer * sizeof(wchar_t));
                    sentence.words = realloc(sentence.words, words_buffer
* sizeof(wchar_t*));
                    words_buffer += BUFFER;
                }
                letters_in_sent++;
                inp_sym = getwchar();
            }
            word[letters_in_word] = L'\0';
            sentence.words[words_in_sent] = word;
            if (words_in_sent == 2) //ищем длину 3 слова
                sentence.len_of_3 = letters_in_word;
            letters_in_word = 0;
            sentence.separator[words_in_sent] = inp_sym;
            if (words_in_sent + 2 == sep_buf){
                sentence.separator = realloc(sentence.separator, sep_buf
* sizeof(wchar_t));
                sep_buf += BUFFER;
```

```
            }
            letters_in_sent++;
            words_in_sent++;
            if (inp_sym != L'.' && inp_sym != L'\n') {
                inp_sym = getwchar();
            }
            else break;
        }
        sentence.num_of_words = words_in_sent;
        words_in_sent = 0;
        flag2 = 0;
        same = 0;
        for (int i = 0; i < num_of_sent; i++){
            struct Sentence prev = inp_text.sentences[i];
            for (int j = 0; j < prev.num_of_words; j++){
                if (prev.num_of_words != sentence.num_of_words)
                    break;
                if (prev.separator[j] == sentence.separator[j])
                    flag2++;
                for (int k = 0; k < wcslen(prev.words[j]); k++)
                    if (towlower(prev.words[j][k]) ==
towlower(sentence.words[j][k])){
                        flag2++;
                    }
            }
            if (flag2 == letters_in_sent) {
                same = 1;
                break;
            }
            flag2 = 0;
        }
        letters_in_sent = 0;
        if (same == 0) {
            inp_text.sentences[num_of_sent] = sentence;
            num_of_sent++;
            if (num_of_sent + 2 == text_buffer) {
                inp_text.sentences = realloc(inp_text.sentences,
text_buffer * sizeof(struct Sentence));
                text_buffer += BUFFER;
            }
        }
    }
    inp_text.len_text = num_of_sent;
    return inp_text;
}

void print_text(struct Text inp_text){
    for (int i = 0; i < inp_text.len_text; i++) {
        for (int j = 0; j < inp_text.sentences[i].num_of_words; j++){
            wprintf(L"%ls%c", inp_text.sentences[i].words[j],
inp_text.sentences[i].separator[j]);
        }
    }
    wprintf(L"\n");
}
```

12. файл text_func.h

```
struct Text get_text();
void print_text(struct Text inp_text);
```