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 С

Студент гр. 1382 _____ Прошичев А.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Прошичев А.В.

Группа 1382

Тема работы: Обработка строк на C

Исходные данные:

Вариант 12

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):

Сделать транслитерацию всех кириллических символов в тексте. Например, подстрока “Какой nice пень” должна принять вид “Kakoj nice pen’” (использовать ГОСТ 7.79-2000)

Для каждого предложения вывести все специальные символы в порядке уменьшения их кода.

Заменить все цифры в тексте их двоичным кодом.

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 22 страниц.

Дата выдачи задания: 15.10.2021

Дата сдачи реферата: 21.12.2021

Дата защиты реферата: 24.12.201

Студент гр. 1382 _____ Прошичев А.В.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

Курсовая работа заключается в реализации программы для обработки текста на языке Си. Для хранения и работы с текстом были использованы структуры и функции стандартных библиотек языка Си.

Программа удаляет повторяющиеся предложения и выводит подсказку с вопросом о дальнейших доступных действиях с текстом. Предусматривается опция выхода из программы. Затем выполняется обработка текста, соответствующая введенной команде и вывод данных. Действия могут выполняться пока пользователь не выйдет из программы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы – создать консольное приложение для обработки текста согласно запросам пользователя. Для выполнения работы необходимо выполнить следующие задачи:

- Ввод, хранение и вывод текста
- Реализация обработки кириллических букв.
- Makefile для сборки программы
- Интерфейс для взаимодействия пользователем
- Взаимодействие с динамической памятью
- Сортировка элементов предложения

Для чтения и хранения текста используется динамическая память, выделяемая и освобождаемая по мере поступления новых данных. Функции выделения памяти и сортировки хранятся в библиотеке *stdlib.h*. Обращение к такой памяти происходит с помощью указателей разных типов. Для комфортной работы с кодом используется *Makefile*. Он даёт возможность разбить программу на отдельные файлы с функциями, чтобы код был не перегружен и читабелен.

Также важной частью программы является обработка кириллических символов. Для работы с ними нужно использовать библиотеки *wchar.h* и *wctype.h*.

1. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

1.1. Ввод текста

При запуске программы пользователю предлагается ввести текст для его обработки. Для хранения текста были реализованы структуры *Sentence* и *Text*. Именно с указателями на их тип работают все остальные функции. Для чтения текста используются *read_snt()* и *read_text()*. По ходу чтения вызывается функция *compare()*, которая сравнивает ново прочитанное предложение и уже сохранённые. Тем самым ещё до занесения в память повторные предложения будут проигнорированы. Также если программа не может больше выделять память, то она завершается, оповестив пользователя об ошибке и освободив всё уже занятую память.

1.2. Выбор операции.

Для успешного взаимодействия пользователя встречает интуитивно понятный текстовой интерфейс. Далее после ввода своего текста пользователь получает инструкцию с номерами команд и их описаниями функций. В зависимости от введённого значения программа выполняет определённые инструкции.

1.3. Первая подзадача

Для транслитиляции символов было создано два статических массива, хранящих алфавит транслированных символов. С помощью циклов функция *translit()* пробегается по каждому символу текста и сравнивает с кириллическими буквами. Если равенство верное, то символ заменяется на свою альтернативную версию.

1.4. Вторая подзадача

Для упрощения реализации подзадачи специальные символы дополнительно сносятся в ещё один символ в функции *read_snt(),* там же они сортируются по возрастанию кода. Сама функция *special_sym()* выводит ответ в определённом формате.

1.5. Третья подзадача

Для поиска цифр в тексте используется функция *iswdigit()* для перевода её в двоичную систему. Далее высчитывает по значению цифры длина кода, при необходимости расширяется количество памяти. Цикл начинается с последнего элемента и сдвигает все элементы до определённой цифры на количество битов в ней. Как только функция *bin_digit()* попадает на индекс, относящийся к двоичной записи цифры, то она начинает записывать остатки от деления на 2. И делая побитовый сдвиг определённого числа на 1 влево.

Под конец такого цикла цифра будет записана в двоичной системе счисления в предложении, а другие его части просто раздвинутся.

1.6. Четвёртая подзадача

Для функции delete_snt() также делается предподсчёт в структуре. Если символ является нечётной цифрой, то структура получает специальную метку odd. Далее функция delete_snt() циклом бежит по предложениям. Если метка на предложении существует, оно полностью удаляется, а на его место сдвигаются другие.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы было создано консольное предложение для обработки текста согласно запросам пользователя. Все задачи также были успешно выполнены: программа может удалять повторяющиеся предложения; транслитерирует все кириллические символы, выводит в порядке возрастания кода специальные символы для каждого предложения, переводит каждую цифру текста в двоичную систему счисления и удаляет предложения, содержащие нечётные цифры. Можно сделать вывод о соответствии полученного результата поставленной цели.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- Сайт (онлайн-справочник) www.c-cpp.ru
- Сайт (онлайн курс) Программирование Си, 1 семестр

ПРИЛОЖЕНИЕ А
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

```
Здравствуйте, уважаемый пользователь! Вас приветствует программа обработки текста.
Сейчас Вам будет предложенно ввести свой текст для обработки, после чего Вы можете выбрать действие из списка.
Предложения следует разделять одним пробелом. Текст считается завершённым после  переноса строки.
-----------------------------------------------------------------------------------------------
Введите свой текст:
Фрукты: яблок, бананы, апельсины, - всё это очень вкусно! (но в этим есть проблемы). ????****&&&:;?;*.
-----------------------------------------------------------------------------------------------
Список возможных действий:
        1: Сделать транслитерацию всех кириллических символов в тексте.
        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.
        0: Завершить обработку.
-----------------------------------------------------------------------------------------------
Введите номер команды: 2
```

Пример 1. Интерфейс.

```
-----------------------------------------------------------------------------------------------
Введите свой текст:
Мои предложения. Специально для них выделено поле.
-----------------------------------------------------------------------------------------------
```

```
Введите номер команды: 1
-----------------------------------------------------------------------------------------------
Полученные результат:
Moi predloženiâ. Special’no dlâ nih vydeleno pole.
-----------------------------------------------------------------------------------------------
```

Пример 2. Первая подзадача.

```
-----------------------------------------------------------------------------------------------
Введите свой текст:
Фрукты: яблок, бананы, апельсины, - всё это очень вкусно! (но в этим есть проблемы). ????****&&&:;?;*.
-----------------------------------------------------------------------------------------------
```

```
Введите номер команды: 2
-----------------------------------------------------------------------------------------------
Полученные результат:
1 предложение: ! ( ) , , , - . :
2 предложение: & & & * * * * * . : ; ; ? ? ? ? ?
-----------------------------------------------------------------------------------------------
```

Пример 3. Вторая подзадача.

```
----------------------------------------------------------------------------------------------------
Введите свой текст:
1. 2. 3. 4. 5.
----------------------------------------------------------------------------------------------------
```

```
Введите номер команды: 3
----------------------------------------------------------------------------------------------------
Полученные результат:
1. 10. 11. 100. 101.
----------------------------------------------------------------------------------------------------
```

Пример 4. Третья подзадача.

```
----------------------------------------------------------------------------------------------------
Введите свой текст:
Предложение с чётной цифрой останется 2468. А все остальные1. Уйдут7. 7. 9. 10.
----------------------------------------------------------------------------------------------------
```

```
Введите номер команды: 4
----------------------------------------------------------------------------------------------------
Полученные результат:
Предложение с чётной цифрой останется 2468.
----------------------------------------------------------------------------------------------------
```

Пример 5. Четвёртая подзадача.

```
Введите номер команды: 0
Приятного Вам дня!
```

Пример 6. Выход из программы.

```
Введите номер команды: e
Неизвестная команда! Проверь наличие такой команды в списке.
```

Пример 7. Неверная команда.

```
----------------------------------------------------------------------------------------------------
Введите свой текст:
1. 3. 51. 755456.
----------------------------------------------------------------------------------------------------
```

```
----------------------------------------------------------------------------------------------------
Полученные результат:
Все предложения были удалены.
----------------------------------------------------------------------------------------------------
```

Пример 8. Пустой текст.

ПРИЛОЖЕНИЕ В

Название файла: main.c

```
#include "main.h"
#include "communicate.h"
#include "tasks.h"
#include "read_text.h"

int main() {
    setlocale(LC_ALL, "");
    wprintf(L"Здравствуйте, уважаемый пользователь! Вас приветствует программа обработки текста.\n");
    wprintf(L"Сейчас Вам будет предложенно ввести свой текст для обработки, после чего Вы можете выбрать действие из списка.\n");
    wprintf(L"Предложения следует разделять одним пробелом. Текст считается завершённым после переноса строки.\n");
    wprintf(L"--------------------------------------------------------------------------------\n");
    wprintf(L"Введите свой текст:\n");
    struct Text *exp = read_text();
    if(exp == NULL) {
        return 0;
    }
    wprintf(L"--------------------------------------------------------------------------------\n");
    wprintf(L"Список возможных действий:\n");
    wprintf(L"\t1: Сделать транслитерацию всех кириллических символов в тексте.\n");
    wprintf(L"\t2: Для каждого предложения вывести все специальные символы в порядке уменьшения их кода.\n");
    wprintf(L"\t3: Заменить все цифры в тексте их двоичным кодом.\n");
    wprintf(L"\t4: Удалить все предложения в которых есть нечетные цифры.\n");
    wprintf(L"\t0: Завершить обработку.\n");
    int action;
    do {
        wprintf(L"--------------------------------------------------------------------------------\n");
        wprintf(L"Введите номер команды: ");
        action = getwchar() - L'0';
        getwchar();
        switch (action) {
            case 1:
                translit(exp);
                wprintf(L"------------------------------------------------------------------------\n");
                wprintf(L"Полученные результат:\n");
                print_text(exp);
                break;
            case 2:
                wprintf(L"------------------------------------------------------------------------\n");
                wprintf(L"Полученные результат:\n");
                special_sym(exp);
```

```
                break;
            case 3:
                bin_digit(exp);
                wprintf(L"------------------------------------------------------------------------
-------------------------\n");
                wprintf(L"Полученные результат:\n");
                print_text(exp);
                break;
            case 4:
                delete_snt(exp);
                wprintf(L"------------------------------------------------------------------------
-------------------------\n");
                wprintf(L"Полученные результат:\n");
                print_text(exp);
                break;
            case 0:
                wprintf(L"Приятного Вам дня!\n");
                break;
            default:
                wprintf(L"Неизвестная команда! Проверь наличие такой команды в списке.\n");
        }
    } while (action);
    clean_memory(exp->text, exp->cnt);
}
```

Название файла: Makefile

```
all: main.o read_text.o tasks.o communicate.o
        gcc main.o read_text.o tasks.o communicate.o -o main
main.o: main.c main.h read_text.h tasks.h communicate.h
        gcc -c main.c
read_text.o: read_text.c main.h read_text.h
        gcc -c read_text.c
tasks.o: tasks.c main.h tasks.h
        gcc -c tasks.c
communicate.o: communicate.c main.h communicate.h
        gcc -c communicate.c
clean:
        rm *.o
```

Название файла: main.h

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

#define MEM_STEP 5

struct Sentence{
    wchar_t *snt;
    int size;
    int cnt;
    wchar_t  *spec_sym;
    int ln;
    int odd;
};

struct Text{
    struct Sentence **text;
    int size;
    int cnt;
};
```

Название файла: tasks.h

```
void translit(struct Text *aim);

void special_sym(struct Text *aim);

int bin_digit(struct Text *aim);

void delete_snt(struct Text *aim);
```

Название файла: tasks.c

```
#include "main.h"
#include "communicate.h"

void translit(struct Text *aim){
    wchar_t *trans_up = L"ABVGDEŽZIJKLMNOPRSTUFHCČŠ”Y’ÈÛÂ";
    wchar_t *trans_lw = L"abvgdežzijklmnoprstufhcčš”y’èûâ";
    for(int idx = 0; idx < aim->cnt; idx++){
        for(int smidx = 0; smidx < aim->text[idx]->cnt; smidx++){
            wchar_t *sym = &aim->text[idx]->snt[smidx];
            for(wchar_t letter = L'A'; letter <= L'Я'; letter++){
                if(letter == *sym) {
                    *sym = trans_up[letter - L'A'];
                    break;
                }
                else if(*sym == towlower(letter)) {
                    *sym = trans_lw[letter - L'A'];
                    break;
                }
                else if(*sym == L'Ë'){
                    *sym = trans_up[6];
                }
                else if(*sym == L'ë'){
                    *sym = trans_lw[6];
                }
            }
        }
    }
}

void special_sym(struct Text *aim){
    for(int idx = 0; idx < aim->cnt; idx++){
        wprintf(L"%ld предложение: ", idx + 1);
        for(int smidx = 0; smidx < aim->text[idx]->ln; smidx++){
            wprintf(L"%lc ", aim->text[idx]->spec_sym[smidx]);
        }
        wprintf(L"%lc", L'\n');
    }
    if(!aim->cnt)
        wprintf(L"Все предложения были удалены.\n");
}

int bin_digit(struct Text *aim){
    for(int idx = 0; idx < aim->cnt; idx++){
        for(int elem = 0; elem < aim->text[idx]->cnt;){
```

```
            if(!iswdigit(aim->text[idx]->snt[elem]) || aim->text[idx]->snt[elem] < L'2') {
                elem++;
                continue;
            }
            aim->text[idx]->odd = 1;
            if(aim->text[idx]->cnt >= aim->text[idx]->size - 4){
                aim->text[idx]->size += MEM_STEP;
                wchar_t *nw = (wchar_t *) realloc(aim->text[idx]->snt, aim->text[idx]->size *
sizeof(wchar_t));
                if(nw == NULL){
                    err(1);
                    clean_memory(aim->text, aim->size);
                    return 1;
                }
                aim->text[idx]->snt = nw;
            }
            int num = aim->text[idx]->snt[elem] - L'0';
            int step, ind;
            if(num < 4){
                step = 1;
            }
            else if(num < 8){
                step = 2;
            }
            else{
                step = 3;
            }
            aim->text[idx]->cnt += step;
            for(ind = aim->text[idx]->cnt; elem + step < ind; ind--){
                aim->text[idx]->snt[ind] = aim->text[idx]->snt[ind - step];
            }
            for(ind; elem <= ind; ind--){
                aim->text[idx]->snt[ind] = (num % 2) + L'0';
                num >>= 1;
            }
            elem += step;
        }
    }
    return 0;
}

void delete_snt(struct Text *aim){
    int cnt = 0;
    for(int idx = 0; idx < aim->cnt;){
        if(aim->text[idx]->odd){
            free(aim->text[idx]->snt);
```

```
            free(aim->text[idx]->spec_sym);
            aim->cnt--;
            cnt++;
        }
        else{
            idx++;
        }
        aim->text[idx] = aim->text[idx + cnt];
    }
}
```

Название файла: communicate.h

```
void err(int a);

void print_text(struct Text *txt);

void clean_memory(struct Sentence **del, int n);
```

Название файла: read_text.h

```
int cmp(const void *a, const void *b); //read_text

char compare(struct Sentence **buf, struct Sentence *nw, int cnt);

struct Sentence *read_snt();

struct Text *read_text();
```

Название файла: communicate.c

```
#include "main.h"

void err(int a){
    switch (a) {
        case 1:
            wprintf(L"Ошибка! Выделение памяти невозможно!\n");
            break;
        case 2:
            wprintf(L"Ошибка! Некорректные входные данные! Пожалуйста, разделяйте предложения
пробелом.\n");
            break;
    }
}

void print_text(struct Text *txt) {
    for (int idx = 0; idx < txt->cnt; idx++) {
        wprintf(L"%ls ", txt->text[idx]->snt);
    }
    if(!txt->cnt)
        wprintf(L"Все предложения были удалены.");
    wprintf(L"\n");
}

void clean_memory(struct Sentence **del, int n){
    // del[i]->snt = **(del + i).snt
    for(int i = 0; i < n; i++){
        free(del[i]->snt);
        free(del[i]->spec_sym);
        free(del[i]);
    }
    free(del);
}
```

Название файла: read_text.c

```
#include "main.h"
#include "communicate.h"

int cmp(const void *a, const void *b){
    return *((const int *)a) - *((const int *)b);
}

char compare(struct Sentence **buf, struct Sentence *nw, int cnt){
    for(int idx = 0; idx < cnt; idx++){
        if(buf[idx]->cnt != nw->cnt)
            continue;
        char flag = 0;
        for(int j = 0; j < buf[idx]->cnt; j++){
            if(towlower(buf[idx]->snt[j]) != towlower(nw->snt[j])){
                flag = 1;
                break;
            }
        }
        if(!flag){
            return 1;
        }
    }
    return 0;
}

struct Sentence *read_snt(){
    int size = MEM_STEP, spec_size = MEM_STEP;
    wchar_t *sp_syms = L"\",{}|[]()+-/%\\;\'.:?<=>_!&*#~^";
    wchar_t *buf = (wchar_t *)malloc(size * sizeof(wchar_t));
    wchar_t *spbuf = (wchar_t *) malloc(spec_size * sizeof(wchar_t));
    int idx = 0, spec_idx = 0, odd = 0;
    wchar_t sym;
    while((sym = getwchar()) != L'.'){
        if(wcschr(L"13579", sym))
            odd = 1;
        if(idx == size - 2){
            size += MEM_STEP;
            wchar_t *nw = (wchar_t *) realloc(buf, size * sizeof(wchar_t));
            if(nw == NULL){
                err(1);
                free(buf);
                free(spbuf);
                return NULL;
            }
```

```
            buf = nw;
        }
        if(spec_idx == spec_size - 2){
            spec_size += MEM_STEP;
            wchar_t *nw = (wchar_t *) realloc(spbuf, spec_size* sizeof(wchar_t));
            if(nw == NULL){
                err(1);
                free(buf);
                free(spbuf);
                return NULL;
            }
            spbuf = nw;
        }
        if(sym == L'\n')
            break;
        buf[idx++] = sym;
        if(wcschr(sp_syms, sym)){
            spbuf[spec_idx++] = sym;
        }
    }
    buf[idx++] = sym;
    spbuf[spec_idx++] = L'.';
    buf[idx] = L'\0';
    spbuf[spec_idx] = L'\0';
    qsort(spbuf, spec_idx, sizeof(wchar_t), cmp);
    struct Sentence *beg = (struct Sentence*) malloc(sizeof(struct Sentence));
    beg->snt = buf;
    beg->cnt = idx;
    beg->size = size;
    beg->spec_sym = spbuf;
    beg->ln = spec_idx;
    beg->odd = odd;
    return beg;
}

struct Text *read_text(){
    int size = MEM_STEP;
    struct Sentence **buf = (struct Sentence **) malloc(size* sizeof(struct Sentence *));
    int idx = 0;
    struct Sentence *snt;
    wchar_t sym;
    do{
        if(size - 2 == idx){
            size += MEM_STEP;
            struct Sentence **nw = (struct Sentence **) realloc(buf, size * sizeof(struct
Sentence *));
```

```
            if(nw == NULL){
                err(1);
                clean_memory(buf, idx);
                return NULL;
            }
            buf = nw;
        }
        snt = read_snt();
        if(snt == NULL){
            err(1);
            wprintf(L"%ld\n", idx);
            clean_memory(buf, idx);
            return NULL;
        }
        else if(snt->snt[snt->cnt - 1] == L'\n'){
            snt->snt[snt->cnt - 1] = L'.';
            buf[idx++] = snt;
            break;
        }
        if(!compare(buf, snt, idx))
            buf[idx++] = snt;
        sym = getwchar();
        if(sym != L'\n' && sym != L' '){
            err(2);
            clean_memory(buf, idx);
            return NULL;
        }
    }while(sym != L'\n');
    struct Text *beg = (struct Text *)malloc(sizeof(struct Text));
    beg->text = buf;
    beg->size = size;
    beg->cnt = idx;
    return beg;
```