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 _____ Охотникова Г.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студентка Охотникова Г.С.

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. Далее программа должна запрашивать у пользователя одно из следующих доступных действий:
1) Для каждой подстроки в тексте, задающей время вида “часы:минуты”, вывести номер предложения в котором она встречается и количество минут до текущего времени.
2) В каждом предложении удалить все заглавные латинские буквы.
3) Отсортировать предложения по уменьшению количеству кириллических букв.
4) Удалить все предложения в которых нет специальных символов.
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. Также должен быть написан Makefile.

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 24.11.2020

Дата сдачи реферата: 27.12.2020

Дата защиты реферата: 29.12.2020

Студентка _____ Охотникова Г.С.

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. Были использованы структуры для хранения тексты. Также был организован выбор команды для обработки текста пользователем. Для сборки программы используется Makefile.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В курсовой работе реализована программа по обработке текста в соответствии с введенной пользователем командой, что осуществлено с помощью оператора множественного выбора swich. Для хранения текста использованы структуры, память выделяется динамически, реализована возможность работы не только с латинскими символами, но и с кириллическими с помощью библиотек wchar.h и wctype.h.

Программа разрабатывалась на операционной системе Linux Mint 20 в интерактивной среде разработки Clion и в текстовом редакторе Vim. Компиляция и линковка осуществлялась с помощью Makefile.

1. ЦЕЛЬ И ЗАДАЧИ РАБОТЫ

Цель: Освоить принципы работы с символьными массивами в языке Си, реализовать программу для обработки текста и применить полученные в ходе первого семестра знания на практике.

Задачи:

- Реализация считывания текста
- Реализация функций по обработке текста
- Реализация работы оператора swich
- Создание Makefile

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Считывание и хранение текста:

Для хранения текста использованы структуры Sentence и Text.

Поля структуры Sentence:

- wchar_t* s — указатель на начало предложения;
- wchar_t* time_s — указатель на начало подстроки вида «часы:минуты»;
- int len_s — длина предложения;
- int count_k — количество кириллических символов в предложении;
- long int diff_t — разница между временем в подстроке вида «часы:минуты» и текущим временем.

Поля структуры Text:

- struct Sentence* newtxt — указатель на массив предложений;
- int len_t — количество предложений в тексте.

Ввод текста осуществлен в функции main. Текст хранится в переменной txt, которая является объектом структуры Text. Ввод текста происходит посимвольно.

Сначала символы записываются в предложение — sent(объект структуры Sentence), при этом пробелы в начале предложения пропускаются и не записываются. Затем происходит посимвольное сравнение считанного предложения с предыдущими. Если данное предложение встречено впервые, то оно записывается в txt. Ввод текста заканчивается символом перевода строки.

Также осуществлено выделение дополнительной памяти, если не хватает памяти, выделенной в начале функции main.

2.2. Решение подзадачи №1.

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

Во вложенном цикле происходит проверка пяти идущих подряд символов предложения: если первый символ является цифрой 0, 1 или 2, второй символ является цифрой, третий — двоеточием, четвертый и пятый — цифрами, то эти пять символов в еще одном цикле записываются в поле time_s. Изначально нулевому элементу данного поля присвоен символ «a» для проверки в операторе множественного выбора.

Затем с помощью функции wcstol полученная строка преобразуется в численный типа данных, то есть отдельной переменной hours присваиваются часы, а переменной minutes присваиваются минуты. Для удобства дальнейших подсчетов часы переводятся в минуты и переменной time_inm присваивается время подстроки в минутах.

В начале функции с помощью функций библиотеки time.h таких, как time() и localtime() получено текущее время. Так как переменная now2 является объектом структуры tm, то при помощи обращения к полям этой структуры, отвечающих за часы и минуты, получаем текущее время в минутах. Данное значение присвоено переменной now_inm.

Происходит сравнение времени подстроки и текущего времени: если текущее время больше, то полю diff_t структуры Sentence присваивается данное значение. Если наоборот, то полю diff_t присваивается 720 минут(12 часов) и модуль разницы текущего времени и времени в подстроке.

В case1 выводятся номера предложений, в которых встречается подстрока искомого вида и разница времени подстроки с текущим временем.

2.3. Решение подзадачи №2.

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

Во вложенном цикле проверяется каждый символ текста с использованием кодировки Unicode. Если код символа лежит в дипазоне от 65 до 90, то в цикле происходит удаление данного символа с помощью сдвига.

Функция возвращает отредактированный текст.

2.4. Решение подзадачи №3.

Решение данной подзадачи реализовано с помощью функции qsort(), функции-компоратора comp_count_letters() и вспомогательной функции count_letters().

Вспомогательная функция получает на вход указатель на объект структуры Sentence. В цикле с помощью кодировки Unicode происходит подсчет кириллических символов в предложении. Если код символа лежит в диапазоне от 1040 до 1103, счетчик увеличивается. Функция возвращает полученное значение, а в case 3 в цикле при вызове данной функции полю count_k присваиваются данные значения. Затем в функции-компораторе происходит обращение к уже заполненному полю.

Таким образом, происходит сортировка по уменьшению кириллических символов в предложениях.

2.5. Решение подзадачи №4.

Для решения данной задачи реализована функция delete_special_symb(), которая получает на вход объект структуры Text.

Изначально переменной fact1 присвоено значение 1, что означает, что в предложении нет специальных символов, и каждый раз в начале цикла оно снова становится единицей. Во вложенном цикле происходит проверка: если символ является специальным(то есть не является буквой, цифрой, пробелом,

запятой или точкой), то значению fact1 присваивается ноль(есть специальные символы). Затем, если fact1 истина(не ноль), то с помощью сдвига удаляется предложение, в котором нет специальных символов.
Функция возвращает обработанный текст.

2.6. Вывод обработанного текста.

Вывод обработанного текста реализован в операторе множественного выбора swich.
В цикле происходит ввод команды с консоли, пока пользователь не введет нулевую команду, которая отвечает за окончание работы программы. Также в case 0 происходит освобождение памяти.
Case 1 отвечает за выполнение первой подзадачи. Здесь же происходит вывод в цикле for.
Case 2 отвечает за выполнение второй подзадачи.
Case 3 отвечает за выполнение третьей подзадачи, то есть за сортировку.
Case 4 отвечает за выполнение четвертой подзадачи.
Case 5 реализует вывод отредактированного текста с помощью цикла for.
Также, если пользователь введет некорректную команду, то на экран будет выведено соответствующее сообщение.

3. ТЕСТИРОВАНИЕ

Вызов утилиты make:

```
gcc -c main.c
gcc -c find_time.c
gcc -c count_letters.c
gcc -c  delete_special_symb.c
gcc main.o find_time.o delete_latin_letter.o count_letters.o
 comp_count_letters.o delete_special_symb.o -o main
```

Запуск исполняемого файла:

```
Здравствуйте!
Введите, пожалуйста, текст, который хотите обработать, закончив свой
 ввод символом перевода строки.
```

Первая подзадача:

```
Здравствуйте!
Введите, пожалуйста, текст, который хотите обработать, закончив свой ввод символом перевода строки.
Начало экзаменационной работы в 12:00. Пожалуйста, не опаздывайте.
Введите, пожалуйста, одну из команд:
0: Выход из программы
1: Вывод номеров предложений, в которых встречается подстрока 'часы: минуты' и количества минут до текущего времени
2: Удаление заглавных латинских букв во всех предложениях
3: Сортировка предложений по уменьшению количеству кириллических букв
4: Удаление предложений, в которых нет специальных символов
5: Вывод отредактированного текста на экран
1
Номер предложения: 0
Минут до текущего времени: 1388
```

Вторая подзадача:

```
Здравствуйте!
Введите, пожалуйста, текст, который хотите обработать, закончив свой ввод символом перевода строки.
Hhello world. It's CcoLld outside.
Введите, пожалуйста, одну из команд:
0: Выход из программы
1: Вывод номеров предложений, в которых встречается подстрока 'часы: минуты' и количества минут до текущего времени
2: Удаление заглавных латинских букв во всех предложениях
3: Сортировка предложений по уменьшению количеству кириллических букв
4: Удаление предложений, в которых нет специальных символов
5: Вывод отредактированного текста на экран
2
5
hello world.t's cold outside.
```

Третья подзадача:

```
Здравствуйте!
Введите, пожалуйста, текст, который хотите обработать, закончив свой ввод символом перевод
а строки.
Последний раз, когда я видел ее, был красным. Небо напоминало похлебку, размешанную и кипя
щую. В некоторых местах оно пригорело. В красноте мелькали черные крошки и катышки перца.
Введите, пожалуйста, одну из команд:
0: Выход из программы
1: Вывод номеров предложений, в которых встречается подстрока 'часы: минуты' и количества
минут до текущего времени
2: Удаление заглавных латинских букв во всех предложениях
3: Сортировка предложений по уменьшению количеству кириллических букв
4: Удаление предложений, в которых нет специальных символов
5: Вывод отредактированного текста на экран
3
5
В красноте мелькали черные крошки и катышки перца.Небо напоминало похлебку, размешанную и
кипящую.Последний раз, когда я видел ее, был красным.В некоторых местах оно пригорело.
```

Четвертая подзадача:

```
Здравствуйте!
Введите, пожалуйста, текст, который хотите обработать, закончив свой ввод символом пе
ревода строки.
Раньше дети играли тут в классики — на улице, похожей на страницы в жирных пятнах. Ко
гда я прибыл, еще слышалось эхо. По мостовой топали ноги. Смеялись детские голос    а
, присоленные улыбками, но разлагались быстро. И вот — бомбы.
Введите, пожалуйста, одну из команд:
0: Выход из программы
1: Вывод номеров предложений, в которых встречается подстрока 'часы: минуты' и количе
ства минут до текущего времени
2: Удаление заглавных латинских букв во всех предложениях
3: Сортировка предложений по уменьшению количеству кириллических букв
4: Удаление предложений, в которых нет специальных символов
5: Вывод отредактированного текста на экран
4
5
Раньше дети играли тут в классики — на улице, похожей на страницы в жирных пятнах.По
мостовой топали ноги.И вот — бомбы.
```

ЗАКЛЮЧЕНИЕ

Разработана программа, которая обрабатывает поступающий на вход текст. Были использованы структуры, функции стандартных библиотек, оператор множественного выбора. Была проделана работа по выделению динамической памяти, по использованию кириллических символов, освоены некоторые принципы работы с библиотекой time.h. Итоговая программа справляется со своей задачей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

1. main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <wchar.h>
#include <locale.h>
#include <wctype.h>
#include <time.h>
#include "structs.h"
#include "find_time.h"
#include "count_letters.h"
#include "comp_count_letters.h"
#include "delete_special_symb.h"
#include "delete_latin_letter.h"


int main() {
    int operation;
    int identic = 0;
    int memory = 100, i;
    wchar_t symb;
    setlocale(LC_ALL, "");
    struct Text txt;
    txt.len_t = 0;
    txt.newtxt = malloc(memory*sizeof(struct Sentense *));
    int fact = 1;

    wprintf(L"Здравствуйте!\n"
            "Введите, пожалуйста, текст, который хотите обработать, "
            "закончив свой ввод символом перевода строки.\n");

    while (fact) {
        struct Sentence sent;
        sent.s = malloc(memory*sizeof(wchar_t));
        memory = memory*2;
        sent.len_s = 0;
        symb = (wchar_t) fgetwc(stdin);

        if (symb == L'\n') {
            fact = 0;
        }

        else {
            while (symb == L' ') {
                symb = (wchar_t) fgetwc(stdin);
            }
            sent.s[sent.len_s++] = symb;
            while (symb != L'.') {
                symb = (wchar_t) fgetwc(stdin);
                sent.s[sent.len_s++] = symb;
            }
```

```
            if (sent.len_s == memory) {
                memory = memory*2;
                sent.s = realloc(sent.s, memory*sizeof(wchar_t));
            }
        }
        if (symb ==  L'.') {
            sent.s[sent.len_s+1] = L'\0';
        }
        for (i = 0; i < txt.len_t; i++) {
            identic = 1;
            for (int j = 0; j < sent.len_s && j < txt.newtxt[i].len_s; j+
+){
                if (towlower(sent.s[j]) != towlower(txt.newtxt[i].s[j]))
{
                    identic = 0;
                }
            }
        }
        if (identic == 0) {
            txt.newtxt[txt.len_t++] = sent;
        }
        if (txt.len_t == memory) {
            memory = memory*2;
            txt.newtxt = realloc(txt.newtxt, memory*sizeof(struct
Sentence *));
        }
    }

    wprintf(L"Введите, пожалуйста, одну из команд:\n"
            "0: Выход из программы\n"
            "1: Вывод номеров предложений, в которых встречается
подстрока 'часы: минуты' и количества минут до текущего времени\n"
            "2: Удаление заглавных латинских букв во всех предложениях\n"
            "3: Сортировка предложений по уменьшению количеству
кириллических букв\n"
            "4: Удаление предложений, в которых нет специальных символов\
n"
            "5: Вывод отредактированного текста на экран\n");
    do {
        wscanf(L"%d", &operation);

        switch (operation) {

            case 0:
                for (int a = 0; a < txt.len_t; a++) {
                    free(txt.newtxt[a].s);
                }
                free(txt.newtxt);
                exit;
                break;
            case 1:
                find_time(&txt);
                for (int b = 0; b < txt.len_t; b++) {
                    if (txt.newtxt[b].time_s[0] != L'a') {
                        wprintf(L"Номер предложения: %d\n", b);
                        wprintf(L"Минут до текущего времени: %ld\n",
txt.newtxt[b].diff_t);
```

```
                    }
                }

                break;
            case 2:
                txt = delete_latin_letter(txt);
                break;
            case 3:
                for (int a = 0; a < txt.len_t; a++) {
                    txt.newtxt[a].count_k =
count_letters(&txt.newtxt[a]);
                }
                qsort(txt.newtxt, txt.len_t, sizeof(struct Sentence),
(int (*)(const void *, const void *))comp_count_letters);
                break;
            case 4:
                txt = delete_special_symb(txt);
                break;
            case 5:
                for (int k = 0; k < txt.len_t; k++) {
                    wprintf(L"%ls", txt.newtxt[k].s);
                }
                wprintf(L"\n");
                break;
            default:
                printf("Данные некорректны");
        }
    } while(operation != 0);
    return 0;
}
```

2. structs.h

```
struct  Sentence {
    wchar_t* s;
    wchar_t* time_s;
    int len_s, count_k;
    long int diff_t;
};

struct Text {
    struct Sentence* newtxt;
    int len_t;
};
```

3.find_time.h

```
void find_time(struct Text *txt);
```

4.find_time.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <wchar.h>
#include <locale.h>
#include <wctype.h>
#include <time.h>
```

```
#include "structs.h"
#include "find_time.h"

void find_time(struct Text *txt) {
    time_t now = time(NULL);
    struct tm* now2 = localtime(&now);
    long int now_inm;
    now_inm = (now2->tm_hour)*60 + now2->tm_min;
    int k;
    long hours, minutes, time_inm;
    wchar_t* end;
    for (int i = 0; i < txt->len_t; i++) {
        k = 0;
        txt->newtxt[i].time_s = malloc(5*sizeof(wchar_t));
        txt->newtxt[i].time_s[0] = L'a';
        for (int i1 = 0; i1 < txt->newtxt[i].len_s-5; i1++) {
            if ((txt->newtxt[i].s[i1] == '0' || txt->newtxt[i].s[i1] ==
'1' || txt->newtxt[i].s[i1] == '2')
                 && iswdigit(txt->newtxt[i].s[i1 + 1]) && txt-
>newtxt[i].s[i1 + 2] == ':'
                 && iswdigit(txt->newtxt[i].s[i1 + 3]) && iswdigit(txt-
>newtxt[i].s[i1 + 4])) {
                for (int j = i1; j < i1 + 5; j++) {
                    txt->newtxt[i].time_s[k++] = txt->newtxt[i].s[j];
                }
                hours = wcstol(txt->newtxt[i].time_s, &end, 10);
                minutes = wcstol(end+1, NULL, 10);
                time_inm = hours*60 + minutes;
                if (now_inm >= time_inm) {
                    txt->newtxt[i].diff_t = now_inm - time_inm;
                }
                else {
                    txt->newtxt[i].diff_t = (time_inm - now_inm) + 720;
                }
            }
        }
    }
}
```

5. delete_latin_letter.h

```
struct Text delete_latin_letter(struct Text txt);
```

6. delete_latin_letter.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <wchar.h>
#include <locale.h>
#include <wctype.h>
#include <time.h>
#include "structs.h"
#include "delete_latin_letter.h"

struct Text delete_latin_letter(struct Text txt) {
    int cod;
    for (int i = 0; i < txt.len_t; i++) {
```

```
        for (int j = 0; j < txt.newtxt[i].len_s; j++) {
            cod = txt.newtxt[i].s[j];
            if (cod >= 65 && cod <= 90) {
                for (int k = j; k < txt.newtxt[i].len_s - 1; k++) {
                    txt.newtxt[i].s[k] = txt.newtxt[i].s[k+1];
                }
                txt.newtxt[i].len_s -= 1;
                j -= 1;
                txt.newtxt[i].s[txt.newtxt[i].len_s] = L'\0';
            }
        }
    }
    return txt;
}
```

7. count_letters.h

```
int count_letters(struct Sentence *sent);
```

8. count_letters.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <wchar.h>
#include <locale.h>
#include <wctype.h>
#include <time.h>
#include "structs.h"
#include "count_letters.h"

int count_letters(struct Sentence *sent) {
    int cod;
    int k = 0;
    for (int i = 0; i < sent->len_s; i++) {
        cod = sent->s[i];
        if (cod >= 1040 && cod <= 1103) {
            k++;
        }
    }
    return k;
}
```

9. comp_count_letters.h

```
int comp_count_letters(struct Sentence* sent1, struct Sentence* sent2);
```

10. comp_count_letters.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <wchar.h>
#include <locale.h>
#include <wctype.h>
#include <time.h>
#include "structs.h"
```

```
#include "comp_count_letters.h"

int comp_count_letters(struct Sentence* sent1, struct Sentence* sent2) {
    return sent1->count_k <= sent2->count_k;
}
```

11. delete_special_symb.h

```
struct Text delete_special_symb(struct Text txt);
```

12.delete_special_symb.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <wchar.h>
#include <locale.h>
#include <wctype.h>
#include <time.h>
#include "structs.h"
#include "delete_special_symb.h"

struct Text delete_special_symb(struct Text txt) {
    int fact1 = 1;
    for (int i = 0; i < txt.len_t; i++) {
        fact1 = 1;
        for (int j = 0; j < txt.newtxt[i].len_s; j++) {
            if (txt.newtxt[i].s[j] != '.' && txt.newtxt[i].s[j] != ',' &&
txt.newtxt[i].s[j] != ' '
            && !(iswalpha(txt.newtxt[i].s[j])) && !
(iswdigit(txt.newtxt[i].s[j]))) {
                fact1 = 0;
            }
        }
        if (fact1) {
            for (int k = i; k < txt.len_t-1; k++) {
                txt.newtxt[k].s = txt.newtxt[k+1].s;
                txt.newtxt[k].len_s = txt.newtxt[k+1].len_s;
            }
            txt.len_t = txt.len_t - 1;
        }
    }
    return txt;
}
```

13. Makefile

```
all: main.o find_time.o delete_latin_letter.o count_letters.o
comp_count_letters.o delete_special_symb.o
     gcc main.o find_time.o delete_latin_letter.o count_letters.o
comp_count_letters.o delete_special_symb.o -o main

main.o: main.c find_time.h delete_latin_letter.h count_letters.h
comp_count_letters.h  delete_special_symb.h
     gcc -c main.c
find_time.o: find_time.c
     gcc -c find_time.c
```

```
delete_latin_letter.o: delete_latin_letter.c
	gcc -c delete_latin_letter.c
count_letters.o: count_letters.c
	gcc -c count_letters.c
comp_count_letters.o: comp_count_letters.c
	gcc -c comp_count_letters.c
delete_special_symb.o: delete_special_symb.c
	gcc -c  delete_special_symb.c
clean:
	rm *.o
```