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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: «Обработка текстовых данных»

Студент гр. 2300 _____ Гаранин Р.А.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2022

Задание

НА КУРСОВУЮ РАБОТУ

Студент Гаранин Р.А.

Группа 2300

Тема работы: Обработка текстовых данных

Исходные данные:

Вариант 17

Программе на вход подается текст (текст представляет собой предложения, разделенные точкой. Предложения - набор слов, разделенные пробелом или запятой, слова - набор латинских букв и цифр. Длина текста и каждого предложения заранее не известна.

Программа должна сохранить этот текст в динамический массив строк и оперировать далее только с ним.

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. Найти в предложениях все даты, записанные в виде “<год> <месяц> <day>” (“1886 Jun 03”) и заменить их на строку, показывающую сколько осталось часов до конца года.
2. Вывести все строки выделив слова на четных позициях красным цветом, а на нечетных зеленым.
3. Удалить все предложения, которые начинаются и заканчиваются на одно и то же слово.
4. Отсортировать предложения по увеличению сумме кодов символов первого слова в предложении.

Все сортировки должны осуществляться с использованием функции стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

Предполагаемый объём пояснительной записки: не менее 9 страниц.

Дата выдачи задания: 20.10.2022

Дата сдачи реферата: 19.12.2022

Дата защиты реферата: 21.12.2022

Студент _____ Гаранин Р.А.

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

Курсовая работа представляет собой реализацию программы для обработки текстовых данных и последующей их вывод в зависимости от выбора условия обработки на языке С.

Текст представляет собой двумерный массив, разделенный по предложениям. Разделителями в предложениях может быть только символы пробела либо запятая, а точка является концом предложения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является освоение подходов к работе с тестовыми данными и последующая реализация программы на языке С для их обработки. Для достижения поставленной цели требуется решить следующие задачи:

1) Реализовать выделение памяти для работы с текстом.

2) Реализовать считывание текста произвольной длины.

3) Освоить методы обработки символьных данных.

4) Освоить способы работы с данными строкового типа при помощи функций стандартной библиотеки.

5) Реализовать задания с курсовой работы в виде отдельных функций.

6) Реализовать очищение памяти при завершении программы.

7) Протестировать готовую программу.

2. ХОД ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Выделение памяти, ввод текста.

В основной функции main начинается создание динамических массивов, а именно создаются двойные указатели на введенный и обработанный текст, под которые изначально не выделена память (т.к. по условия курсовой работы, длина текста и каждого предложения заранее не известна).

Далее с помощью функции processing_of_text реализуется ввод текста. Внутри функции реализован цикл while, который кончается при нажатии кнопки ENTER (а именно, при вводе символа переноса строки). Внутри самого цикла при нехватке памяти выделяться дополнительная с помощью функции realloc.

2.2. Обработка и удаление одинаковых предложений.

После ввода текста, он обрабатывается с помощью функции processing_of_text. Внутри данном функции с помощью циклов for реализована обработка текста и разбиение его на отдельные предложения в двумерный массив, для удобства дальнейшей работы с ним.

После обработки текста, вызывается следующая функция, которая также обрабатывает текст, в нашем случае уже двумерный массив. Функция correct_the_text, с помощью двух циклов for и двух условных операторов, проходит по всему массиву и ищет одинаковые предложения, если таковое нашлось, то в его начало ставить “\0”. В конце функции с помощью цикла for и условного оператора реализована очистка массива от ненужных строк в массиве, как итог, функция удаляет повторяющиеся предложения.

2.3. Реализация меню.

Меню программы реализуется в основной функции main. Для начала пользователю выводится функционал программы, среди вариантов пользователь должен выбрать один из них введя в консоль соответствующую цифру. Данное число записывается в переменную и с помощью оператора switch идет исполнение выбранной задачи.

2.4. Подзадача №1.

Первая подзадача представлена в функции dates_in_text и представляет собой поиск в предложениях все даты, записанные в виде “<год> <месяц> <day>” (“1886 Jun 03”) и замена их на строку, показывающую сколько осталось часов до конца года. Как итог, программа выводит строку с изменённой датой соответственно.

2.5. Подзадача №2.

Вторая подзадача представлена в функции marker_for_text и представляет собой вывод всех строки, выделив слова на четных позициях красным цветом, а на нечетных зеленым. В данной функции с помощью двух циклов for и нескольких условных операторов идет поочередный вывод и окрашивание элементов массива, в зависимости от его расположения.

2.6. Подзадача №3.

Третья подзадача представлена в функции del_of_predl и представляет собой - удаление предложения, которые начинаются и заканчиваются на одно и то же слово. В данной функции мы сначала копируем строчку в переменную и находим в ней первое слово и также копируем первую строчку, но уже в другую переменную и находим в нем последнее слово. Далее мы проверяем их, как итог, функция удаляет предложение если две эти переменные одинаковы, в противном случае идем дальше. В конце функции выводим обработанный текст.

2.7. Подзадача №4.

Четвертая подзадача представлена в функции sorting_by_code и представляет собой - сортировку предложений по увеличению сумме кодов символов первого слова в предложении. В данной функции с помощью функции qsort реализована сортировка предложений от меньшего к большему. В конце функции с помощью цикла for реализован вывод получившегося массива (нашего текста).

2.8. Очистка памяти.

Для очистки памяти в конце программы реализована функция free_up_memory, принимающая на вход текст и количество его предложений. С

помощью цикла for очищает память для каждого предложения и текста, как итог функция полностью очищает память после работы нашей программы.

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы было изучено не только работа с циклами и условными операторами, но и динамическое выделение памяти и ее изменение, а также обработка текстовых данным с помощью функций на языке С.

Как итог, была написана программа, которая работает полностью согласно заданиям к курсовой работе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Керниган Б., Ритчи Д. Язык программирования Си. СПб.: "Невский Диалект", 2001. 352 с.

Приложение А
ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

ВЫПОЛНЕНИЕ 1 ЗАДАЧИ

```
Введите предложения. Для конца ввода нажмите ENTER.
SD sdfs fds fs dsf 2000 Dec 20. IJ jij ijsd fsf sdf g.
Вы можете выбрать одно из следующих действий:
1) Найти в предложениях ce даты записанные в виде "<год> <месяц> <day>" ("1886 Jun 03") и заменить их на строку показывающую сколько ост►лось часов до конца года.
2) Вывести все строки выделив слова на четных позициях красным цветом, а на нечетных зеленым.
3) Удалить все предложения, которые начинаются и заканчиваются на одно и то же слово.
4) Отсортировать предложения по увеличению сумме кодов символов первого слова в предложении.
5) Выход их программы
Введите вариант действия: 1
Выбрано действие 1.
BSD sdfs fds fs dsf 288.
IJ jij ijsd fsf sdf g.
```

ВЫПОЛНЕНИЕ 2 ЗАДАЧИ

```
Введите предложе►ия. Для конца ввода нажмите ENTER.
asd as dasd as da sdas dad. sd fsd fsd fs rtg. dfg d gd hg.
Вы можете выбрать одно из следующих действий:
1) Найти в предложениях ce даты записанные в виде "<год> <месяц> <day>" ("1886 Jun 03") и заменить их на строку показывающую сколько ост►лось часов до конца года.
2) Вывести все строки выделив слова на четных позициях красным цветом, а на нечетных зеленым.
3) Удалить все предложения, которые начинаются и заканчиваются на одно и то же слово.
4) Отсортировать предложения по увеличению сумме кодов символов первого слова в предложении.
5) Выход их программы
Введите вариант действия: 2
Выбрано действие 2.
asd as dasd as da sdas dad.
sd fsd fsd fs rtg.
dfg d gd hg.
```

ВЫПОЛНЕНИЕ 3 ЗАДАЧИ

```
Введите предложения. Для конца ввo►a нажмите ENTER.
dfg dfg d gdfg dfgd g. sdh d fhr th rh sdh. dfga dh.
Вы можете выбрать одно из следующих действий:
1) Найти в предложениях ce даты записанные в виде "<год> <месяц> <day>" ("1886 Jun 03") и заменить их на строку показывающую сколько ост►лось часов до конца года.
2) Вывести все строки выделив слова на четных позициях красным цветом, а на нечетных зеленым.
3) Удалить все предложения, которые начинаются и заканчиваются на одно и то же слово.
4) Отсортировать предложения по увеличению сумме кодов символов первого слова в предложении.
5) Выход их программы
Введите вариант действия: 3
Выбрано действие 3.
dfg dfg d gdfg dfgd g.
dfga dh.
```

ВЫОПЛНЕНИЕ 4 ЗАДАЧИ

```
Введите предложения. Для конца ввo►a нажмите ENTER.
dfg dfgd gdfgd fg dfg dg dgd. Sfdf gdfgd fgd fh df. E dfh dghd dfg.
Вы можете выбрать одно из следующих действий:
1) Найти в предложениях ce даты записанные в виде "<год> <месяц> <day>" ("1886 Jun 03") и заменить их на строку показывающую сколько ост►лось часов до конца года.
2) Вывести все строки выделив слова на четных позициях красным цветом, а на нечетных зеленым.
3) Удалить все предложения, которые начинаются и заканчиваются на одно и то же слово.
4) Отсортировать предложения по увеличению сумме кодов символов первого слова в предложении.
5) Выход их программы
Введите вариант действия: 4
Выбрано действие 4.
E dfh dghd dfg.
dfg dfgd gdfgd fg dfg dg dgd.
Sfdf gdfgd fgd fh df.
```

ВЫПОЛНЕНИЕ 5 ЗАДАЧИ

```
Введите предложения. Для конца ввo►a нажмите ENTER.
fg d gd fg dfg d gd. df gd fgd g dgd fg. df gd fgd f.
Вы можете выбрать одно из следующих действий:
1) Найти в предложениях ce даты записанные в виде "<год> <месяц> <day>" ("1886 Jun 03") и заменить их на строку показывающую сколько ост►лось часов до конца года.
2) Вывести все строки выделив слова на четных позициях красным цветом, а на нечетных зеленым.
3) Удалить все предложения, которые начинаются и заканчиваются на одно и то же слово.
4) Отсортировать предложения по увеличению сумме кодов символов первого слова в предложении.
5) Выход их программы
Введите вариант действия: 5
Выбрано действие 5.
Всего хорошег►)
```

Приложение Б
ИСХОДНЫЙ КОД ПРОГРАММЫ

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

char* input_mass(int *countPR){
    int M = 10;
    int N = 10;
    char c;
    int i = 0;
    int new = 0;

    char *arr = malloc(N * sizeof(char));

    while(1){
        c = getchar();
        if(c == '\n'){
          arr[i] = '\0';
          break;
        }
        if(new == 1){
            new = 0;
            while((c = getchar()) == ' ');
        }
        if(i+1 == N){
            N = N + 10;
            arr = realloc(arr, N * sizeof(char));
        }
        arr[i] = c;
        i++;
```

```
        if(c =='.'){
            (*countPR)++;
            new = 1;
        }
    }

    return arr;
}

void processing_of_text(char ***mass, char **str, int countPR){
    int M = 10;
    int F = 0;
    int G = 0;
    int count1 = 0;

    for(int i = 0; i < countPR; i++){
        (*mass)[i] = malloc(M * sizeof(char));
    }

    for(int i = 0; (*str)[i]; i++){
        if(G+1 == M){
            M = M + 10;
            (*mass)[F] = realloc((*mass)[F], M * sizeof(char));
        }
        (*mass)[F][G] = (*str)[i];
        G++;
        if((int)(*str)[i] == 46){
            (*mass)[F][G] = '\0';
            F++;
            G = 0;
            M = 10;
        }
    }
}
```

```
int compare(char *str1, char *str2){
    int first = strlen(str1);
    int second = strlen(str2);

    if(first != second){
        return 0;
    }

    int identical = 1;
    for(int i = 0; i < first; i++){
        if(str1[i] != str2[i]){
            identical = 0;
        }
    }

    return identical;
}

void correct_the_text(char ***mass, int *countPR){
    for(int i = 1; i < *countPR; i++){
        for(int j = 0; j < i; j++){
            if(strlen((*mass)[i]) == strlen((*mass)[j])){
                if(compare((*mass)[i], (*mass)[j])){
                    (*mass)[i][0] = '\0';
                    break;
                }
            }
        }
    }

    int j = 0;
    for(int i = 0; i < *countPR; i++){
        if((*mass)[i][0] == '\0'){
```

```
            free((*mass)[i]);
            j++;
            continue;
        }
        (*mass)[i - j] = (*mass)[i];
    }
    *countPR -= j;
}

void dates_in_text(char ***mass, int countPR){
    for(int i = 0; i < countPR; i++){
        int n = 0;
        int v = 0;
        int otvet;
        char *t1;
        char *t2;
        char *t3;
        int dr = 0;
        int NM;
        char mes[12][4] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
"Aug", "Sep", "Oct", "Nov", "Dec"};

        char *copy1 = malloc(strlen((*mass)[i]) * sizeof(char));
        strncpy(copy1, (*mass)[i], strlen((*mass)[i]) * sizeof(char));

        char *temp = strtok(copy1, " ,.");
        while(temp){
            if(n == 2){
                t3 = temp;
                for(int i = 0; temp[i]; i++){
                    if(isdigit(temp[i])){
                        n = 3;
                        continue;
                    }else{
```

```
            n = 0;
            break;
        }
    }
}

if(n == 1){
    for(int w = 0; w < 12; w++){
        if(strlen(temp) == 3 && memcmp(temp, mes[w], 3) == 0){
            NM = w+1;
            n = 2;
            t2 = temp;
            break;
        }else{
            n = 0;
        }
    }
}

if(n == 0){
    t1 = temp;
    for(int q = 0; temp[q]; q++){
        if(isdigit(temp[q])){
            n = 1;
            continue;
        }else{
            n = 0;
            break;
        }
    }

    dr = t1 - copy1;
}
if(n == 3){
```

```
        int WS = atoi(t3);

        time_t time_of_day;
        struct tm tmbuf, *tptr;
        time_of_day = NULL;
        memset(&tmbuf, 0, sizeof(tmbuf));
        tmbuf.tm_mday = WS;
        tmbuf.tm_mon = NM;
        tmbuf.tm_mon--;
        tmbuf.tm_year = 2010 - 1900;
        time_of_day = mktime(&tmbuf);
        otvet  = (365 - tmbuf.tm_yday) * 24;
        break;
    }
    temp = strtok(NULL, " ,.");

}

if(n == 3){
    int len = strlen(t1) + 7;
    int ln = 0;
    char *che = malloc(5 * sizeof(char));
    che[0] = '0';
    char c;
    int gh = 0;
    while(otvet > 0){
        che[ln++] = otvet%10 + '0';
        otvet/=10;
    }
    for(gh = 0; gh < (ln/2); gh++){
        c = che[gh];
        che[gh] = che[ln - 1 - gh];
        che[ln - 1 - gh] = c;
    }
```

```
                for(int y = dr; y < (dr + ln); y++){
                    (*mass)[i][y] = che[y - dr];
                }

                memmove((*mass)[i] + dr + ln, (*mass)[i] + dr + len,
(strlen((*mass)[i]) - dr - len + 1) * sizeof(char));
            }
        }
        for(int err = 0; err < countPR; err++){
            puts((*mass)[err]);
        }
    }

    void marker_for_text(char **mass, int countPR){
        int num;
        for(int i = 0; i < countPR; i++){
            num = -1;
            for(int j = 0; mass[i][j]; j++){
                if(isalnum(mass[i][j]) && (num == -1 || num == 1)){
                    num = 1;
                    printf("\033[32m%c\033[0m", mass[i][j]);
                }
                else if(isalnum(mass[i][j]) && (num == -2 || num == 2)){
                    num = 2;
                    printf("\033[31m%c\033[0m", mass[i][j]);
                }
                else{
                    if(num == 1){
                        num = -2;
                    }
                    if(num == 2){
                        num = -1;
                    }
                    printf("%c", mass[i][j]);
```

```
            }
        }
        printf("\n");
    }
}

void del_of_predl(char ***mass, int *countPR){
    for(int i = 0; i < *countPR; i++){

        char *copy1 = malloc(strlen((*mass)[i]) * sizeof(char));
        strncpy(copy1, (*mass)[i], strlen((*mass)[i]) * sizeof(char));
        char *first = strtok(copy1, " .,");

        char *second = malloc((strlen(first) + 1) * sizeof(char));
        strncpy(second, &(*mass)[i][strlen((*mass)[i]) - strlen(first) - 1],
strlen(first) * sizeof(char));
        if(strlen((*mass)[i]) - strlen(first) - 1 != 0 &&
isalpha((*mass)[i][strlen((*mass)[i]) - strlen(first) - 2])){
            continue;
        }
        second[strlen(first)] = '\0';
        if(strcmp(first, second) == 0){
            (*mass)[i][0] = '\0';
        }
    }

    int j = 0;
    for(int i = 0; i < *countPR; i++){
        if((*mass)[i][0] == '\0'){
            free((*mass)[i]);
            j++;
            continue;
        }
        (*mass)[i - j] = (*mass)[i];
```

```
    }
    *countPR -= j;

    for(int i = 0; i < *countPR; i++){
        puts((*mass)[i]);
    }
}

int cmp(const void** s1, const void** s2){
    char *str1 = (char*)*s1;
    char *str2 = (char*)*s2;

    char *copy1 = malloc(strlen(str1) * sizeof(char));
    strncpy(copy1, str1, strlen(str1) * sizeof(char));
    char *temp1 = strtok(copy1, " .,");
    int sum1 = 0;
    for(int i = 0; temp1[i]; i++){
        sum1+=(int)temp1[i];
    }

    char *copy2 = malloc(strlen(str2) * sizeof(char));
    strncpy(copy2, str2, strlen(str2) * sizeof(char));
    char *temp2 = strtok(copy2, " .,");
    int sum2 = 0;
    for(int i = 0; temp2[i]; i++){
        sum2+=(int)temp2[i];
    }

    if(sum1 > sum2){
        return 1;
    }else if(sum1 == sum2){
        return 0;
    }else{
        return -1;
```

```
    }
}

void sorting_by_code(char ***mass, int countPR){
    qsort(*mass, countPR, sizeof(char*), cmp);
    for(int i = 0; i < countPR; i++){
        puts((*mass)[i]);
    }
}

void free_up_memory(char **mass, char *str, int countPR){
    free(str);
    for(int i = 0; i < countPR; i++){
        free(mass[i]);
    }
    free(mass);
}
//iaflhkgd dgdfg, fsdgs sdfgs. sdf, df hj ds, bv ef hj asd uky, cxzv. dthrt sdvs, ghjg asd rty sdv. sfghd. dgh. dgh. sdf, fsg.
int main(){
    int countPR = 0;

    setlocale(LC_ALL, "Rus");

    printf("Введите предложения. Для конца ввода нажмите ENTER.\n");
    char *str = input_mass(&countPR);
    char **mass = malloc(countPR * sizeof(char*));
    processing_of_text(&mass, &str, countPR);
    correct_the_text(&mass, &countPR);

    printf("Вы можете выбрать одно из следующих действий:\n");
    printf("1) Найти в предложениях все даты записанные в виде “<год> <месяц> <day>” (“1886 Jun 03”) и заменить их на строку показывающую сколько осталось часов до конца года.\n");
```

```
        printf("2) Вывести все строки выделив слова на четных позициях красным цветом, а на нечетных зеленым.\n");
        printf("3) Удалить все предложения, которые начинаются и заканчиваются на одно и то же слово.\n");
        printf("4) Отсортировать предложения по увеличению сумме кодов символов первого слова в предложении.\n");
        printf("5) Выход их программы\n");

        int variant = 0;
        printf("Введите вариант действия: ");
        scanf("%d", &variant);

        switch(variant){

        case 1:{
           printf("Выбрано действие 1.\n");
           dates_in_text(&mass, countPR);
           break;
        }

        case 2:{
           printf("Выбрано действие 2.\n");
           marker_for_text(mass, countPR);
           break;
        }

        case 3:{
           printf("Выбрано действие 3.\n");
           del_of_predl(&mass, &countPR);
           break;
        }

        case 4:{
           printf("Выбрано действие 4.\n");
```

```
            sorting_by_code(&mass, countPR);
            break;
        }

        case 5:{
            printf("Выбрано действие 5.\n");
            printf("Всего хорошего:)");
            break;
        }

        default:
            printf("Введенные данные не корректны! Попробуйте еще раз!\n");
            break;
        }

        free_up_memory(mass, str, countPR);

        return 0;
}
```