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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Обработка текста

Студент гр. 1381 _____ Дудко М.А

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2022

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Дудко М.А.

Группа 1381

Тема работы: Обработка текста

Исходные данные: 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.

Содержание пояснительной записки:
Разделы «Содержание», «Введение», «Заключение», «Задание работы», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 15.10.2021

Дата сдачи реферата: 26.02.2022

Дата защиты реферата: 01.03.2022

Студент _____ Дудко М.А.

Преподаватель _____ Жангиров Т.Р.

АННОТАЦИЯ

В ходе выполнения курсовой работы была написана программа, соответствующая всем требованиям поставленных задач. Был разработан интерфейс, взаимодействующий с пользователем. Все задачи были выполнены и вынесены в отдельные функции. Функции обрабатывают текст в выводе.

SUMMARY

In the course of the course work, a program was written that meets all the requirements of the tasks. An interface that interacts with the user has been developed. All tasks have been completed and put into separate functions. The functions process the text in the output.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы:

На языки С написать программу для обработки текста, состоящий из латинских букв, цифр, знаков и специальных символов.

Основные задачи:

Научиться работать с текстом на языке С, использовать стандартные библиотеки для работы со строками. Разбить каждую подзадачу на отдельную функцию. Эффективная работа с динамической памятью компьютера.

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 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 во всем тексте даты записанные в подстроке вида “d<day>m<month>y<year>” и вывести все даты по возрастанию в формате “DD:MM:YYYY:. Пример даты в тексте, “d14m03y0988”.
2. Удалить все предложения в которых количество слов нечетно.
3. Преобразовать все слова в которых нет цифр так, чтобы все буквы кроме последней были прописными.
4. Вывести все предложения в которых нет заглавных букв.

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.

2. ХОД РАБОТЫ

2.1. Создание функции ввода предложений read_sentence

Функция посимвольно считывает введенный текст в динамический массив, динамическая память для которого была выделена с помощью функции malloc, пока не будет введен символ «.» или перевод строки. Так же при необходимости дополнительно выделяется 5 байт памяти при помощи функции realloc. Функция возвращает предложение.

2.2. Создание функции ввода текста read_text

```
struct Sentence *read_sentence() {
    int size = MEM_STEP;
    char *buf = malloc( size: size * sizeof(char));
    char temp = getchar();
    int n = 0;
    do {
        if (n >= size - 2) {
            char *t = realloc( ptr: buf,  size: size + MEM_STEP);
            if (!t) {
                return NULL;
            }
            size += MEM_STEP;
            buf = t;
        }

        buf[n] = temp;
        temp = getchar();
        n++;

    } while (temp != '\n' && temp != '.');

    buf[n] = temp;
    buf[n + 1] = '\0';
    if (buf[0] == '\n') {
        buf++;
    }
```

Данная функция формирует текст из предложений полученных при помощи прошлой функции. Выделение и добавление памяти проводятся

аналогично, так же удаляются повторяющееся предложения при помощи функции unigue_sent

```
struct Text read_text() {
    int size = MEM_STEP;
    struct Sentence **text = malloc( size: size * sizeof(struct Sentence *));
    struct Sentence *temp;
    int n = 0;
    int nlcount = 0;

    do {
        temp = read_sentence();

        if (n >= size - 2) {
            struct Sentence **t = realloc( ptr: text,  size: (size + MEM_STEP) * sizeof(struct Sentence *));

            if (!t) {
                puts( s: "Allocation error");
            }

            text = t;
            size += MEM_STEP;
        }
        if (temp->str[0] == '\n' && temp->str[1] == '\0') {
            nlcount++;

        } else {
            while (temp->str[0] == '\t' || temp->str[0] == ' ' || temp->str[0] == '\n') {
                temp->str++;
            }
            nlcount = 0;
            if (unique_sent( txt: text,  sent: temp, n)) {
                text[n] = temp;
                n++;
            }
        }
    } while (nlcount < 1);

    struct Text txt;
    txt.text = text;
    txt.size = size;
    txt.n = n;
read_text
```

2.3.Функция первичной обработки unique_sent

Функция проверяет предложения на повторяющиеся, считая повторения посимвольно, если предложение символ в символ равны, то соответственно сами предложения тоже равны.

```
int unique_sent(struct Sentence **txt, struct Sentence *sent, int n) {
    for (int i = 0; i < n; i++) {
        int k = 0;
        for (int j = 0; j < strlen( s: sent->str); j++) {
            if (toupper( c: txt[i]->str[j]) == toupper( c: sent->str[j]))
                k++;

        }
        if (k == strlen( s: sent->str) && k == strlen( s: txt[i]->str))
            return 0;

    }
    return 1;
}
```

2.4.Первая подзадача

Функция func_date ищет в тексте слово соответствующее шаблону, а затем с помощью функции strtok вырезает из копии предложения нужны значения дня, месяца и года. Затем с помощью функция atoi приобразует в число и записывает в массив структур. Функция cmp — компаратор, сравнивает даты, и выясняет какая больше. Затем массив структур сортируется по возрастанию с помощью функции qsort

```
int unique_sent(struct Sentence **txt, struct Sentence *sent, int n) {
    for (int i = 0; i < n; i++) {
        int k = 0;
        for (int j = 0; j < strlen( s: sent->str); j++) {
            if (toupper( c: txt[i]->str[j]) == toupper( c: sent->str[j]))
                k++;

        }
        if (k == strlen( s: sent->str) && k == strlen( s: txt[i]->str))
            return 0;

    }
    return 1;
}
```

```c
void func_data(struct Text text){ // func 1
    struct Data *data = malloc( size: sizeof(struct Data)*100);
    int count = 0;
    int p = 0;
    for(int i = 0; i < text.n; i++){
        char strcopy[strlen( s: text.text[i]->str)];
        strcpy( dest: strcopy,  src: text.text[i]->str);
        char temp;
        for(int j = 0; j < strlen( s: text.text[i]->str); j++){
            temp = text.text[i]->str[j];
            if (temp == 'd' && isdigit(text.text[i]->str[j+1]) && isdigit(text.text[i]->str[j+2]) &&
            (text.text[i]->str[j+3]) == 'm' && isdigit(text.text[i]->str[j+4]) && isdigit(text.text[i]->str[j+5]) &&
            text.text[i]->str[j+6] == 'y' && isdigit(text.text[i]->str[j+7]) && isdigit(text.text[i]->str[j+8]) &&
            isdigit(text.text[i]->str[j+9]) && isdigit(text.text[i]->str[j+10]) ){
                char* d = strtok( s: strcopy + j + 1,  delim: "m");
                int int_d = atoi( nptr: d);

                char* m = strtok( s: strcopy + j + 4,  delim: "y");
                int int_m = atoi( nptr: m);

                char* y = strtok( s: strcopy + j + 7,  delim: " .,");
                int int_y = atoi( nptr: y);

                data[p].year = int_y;
                data[p].month = int_m;
                data[p].day = int_d;
                count++;
                p++;
            }
            }
        }
    qsort( base: data,  nmemb: count, size: sizeof(struct Data), compar: cmp);

    for(int g = 0; g < count; g++){
        printf( format: "%02d%c%02d%c%04d\n", data[g].day,':',data[g].month,':',data[g].year);
    }
    free( ptr: data);
    }
```

2.5.Вторая подзадача

Функция delete_odd_count посимвольно проходится по тексту и находит индекс начала и конца слова, а так же считает их количество в предложении, если их нечетное количество то предложение удаляется с помощью функции memmove. Ячейки памяти хранящие следующее предложения перемещаются на место текущего.

```
void delete_odd_count(struct Text text) { // 2 func
    int k = 0;
    while (k < text.n) {
        int i = 0;
        int s = 0;
        char temp;
        while (i < strlen( s: text.text[k]->str)) {
            temp = text.text[k]->str[i];
            i++;
            if (temp == '\n' || temp == '.' || temp == '!' || temp == ' ') continue;
            i--;
            int j = i;
            do {
                j++;
                temp = text.text[k]->str[j];
            } while (temp != '\n' && temp != '.' && temp != '!' && temp != ' ' && temp != ',');
            s++;
            i = j;
        }

        if (s % 2 == 1) {
            memmove( dest: text.text + k,  src: text.text + k + 1, n: (text.n - k - 1) * sizeof(*text.text));
            text.n--;
        } else k++;
    }
}
```

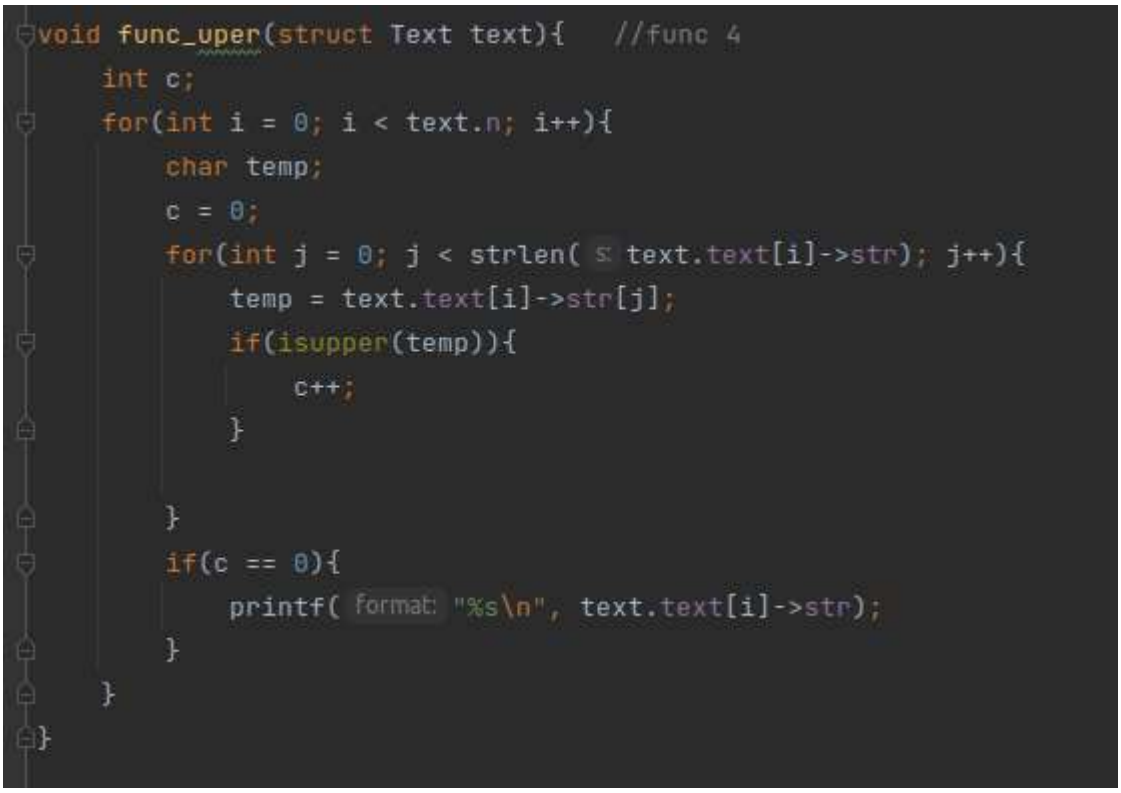
2.6.Третья подзадача

Функция func_tolower аналогично находит находит начало и конец слова, так же проверяет на наличие цифр и записывает длину слова, затем если в нем цифры отсутствует то при помощи цикла приводит к нижнему регистру все слово кроме последнего символа.

```
void func_tolower(struct Text text) { //3 func
    int k = 0;
    while (k < text.n) {
        int i = 0;
        char temp;
        while (i < strlen( s: text.text[k]->str)) {
            temp = text.text[k]->str[i];
            i++;
            if (temp == '\n' || temp == '.' || temp == '!' || temp == ' ') continue;
            i--;
            int j = i;
            int count = 0;
            int count_digit = 0;
            do {
                j++;
                temp = text.text[k]->str[j];
                if (isdigit(temp)) {
                    count_digit++;
                }
                count++;
            } while (temp != '\n' && temp != '.' && temp != '!' && temp != ' ' && temp != ',');
            if (count_digit == 0){
                for (int p = 0; p < count - 1; p++) {
                    text.text[k]->str[i] = tolower( c: text.text[k]->str[i]);
                    i++;
                }
            }
            i = j;
        }
        k++;
    }
}
```

2.7.Четвертая подзадача

Функция func_uper посимвольно проходится по предложению, и проверяет на наличие верхнего регистра в предложении, если его не находит, то выводит предложение.



```
void func_uper(struct Text text){   //func 4
    int c;
    for(int i = 0; i < text.n; i++){
        char temp;
        c = 0;
        for(int j = 0; j < strlen( s: text.text[i]->str); j++){
            temp = text.text[i]->str[j];
            if(isupper(temp)){
                c++;
            }

        }
        if(c == 0){
            printf( format: "%s\n", text.text[i]->str);
        }
    }
}
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были освоены механизмы работы со структурами, двумерными и одномерными массивами структур. Выделение и освобождение динамической памяти. Правильная структуризация кода и разбиение отдельных задач на отдельные функции. Освоена библиотека функций для работы со строками и не только.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. КЕРНИГАН Б. В., РИТЧИ Д. М. ЯЗЫК ПРОГРАММИРОВАНИЯ СИ: ПЕР. С АНГЛ. — 3-Е
ИЗД. — СПБ.: НЕВСКИЙ ДИАЛЕКТ, 2001. — 352 С.
2. HTTPS://EN.CPPREFERENCE.COM/W/C/STRING
3. HTTP://WWW.C-CPP.RU