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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Программирование»

Тема: Указатели и массивы

Студентка гр. 2384 _____ Валеева А. А.

Преподаватель _____ Гаврилов А. В.

Санкт-Петербург

2022

Цель работы.

Изучение указателей и динамического выделения памяти для массивов, освоение работы со строками и двумерными массивами в языке С.

Задание.

Вариант № 5.

Напишите программу, которая форматирует некоторый текст и выводит результат на консоль.

На вход программе подается текст, который заканчивается предложением "Dragon flew away!".

Предложение (кроме последнего) может заканчиваться на:

- . (точка)
- ; (точка с запятой)
- ? (вопросительный знак)

Программа должна изменить и вывести текст следующим образом:

- Каждое предложение должно начинаться с новой строки.
- Табуляция в начале предложения должна быть удалена.
- Все предложения, в которых больше одной заглавной буквы, должны быть удалены.
- Текст должен заканчиваться фразой "Количество предложений до n и количество предложений после m", где n - количество предложений в изначальном тексте (**без учета** терминального предложения "Dragon flew away!") и m - количество предложений в отформатированном тексте (без учета предложения про количество из данного пункта).

*** Порядок предложений не должен меняться**

*** Статически выделять память под текст нельзя**

*** Пробел между предложениями является разделителем, а не частью какого-то предложения.**

Выполнение работы.

В начале подключаем библиотеки *stdio.h*, *string.h*, *stdlib.h* и *ctype.h*, а также создаем макрос: *define TEXT_SIZE*.

Функция *readSentence()* получает посимвольно предложение и выделяет для него память с помощью функции *malloc*, размер выделяемой памяти равен *size* — переменной, изначально равной *TEXT_SIZE.* Далее с помощью функции *getchar* считываем символы. Удаляем табуляцию в начале предложения следующим образом: если полученный символ не является пробелом или \t, то тогда мы его вносим в строку. Далее добавляем символы в строку, пока не встретим знак препинания, означающий конец строки («.», «;», «?»). Не забываем проверять, не превысила ли длина строки *sentence_len*
выделенный размер *size*, если нужно — выделяем еще память и меняем с помощью функции *realloc*. Если встречаем «.», «;», «?», то последним символом в строку вносим \0. В конце возвращаем предложение - *return sentence.*

Следующая функция — *check_sentence*, аргументом которой является указатель на начало предложения. Она помогает проверить, какое количество заглавных букв содержится в передаваемой строке. Вводим переменную *count* для подсчета количества. Циклом *for* проходим посимвольно до конца строки. С помощью метода *issupper* проверяем, является ли символ заглавной буквой, если да — то *count* увеличиваем на 1. В конце выводим 1, если строка содержит менее 2 заглавных букв в строке, 0 в обратном случае.

Последняя функция — *main()*. Для начала введем в ней следующие переменные — *size*, равная константе *TEXT_SIZE, end_sent = «Dragon flew away!»* - конечное предложение текста*, sentence_add* — переменная для подсчета тех строк, которые нам подходят (они будут выведены на экран) и *sentence_delete* — количество неподходящих строк. Выделяем с помощью функции *malloc* память для хранения *text* - массива указателей размером *size.* Получаем первое предложение с помощью вызова функции *readSentence(),*

проверяем его. Проверка будет подробнее расписана дальше. Далее используем цикл *while*, условие которого — пока полученная и конечная(*end_sent*) строки не совпадут. Это условие проверяем с помощью метода *strcmp*, который посимвольно сравнивает строки и выводит 0, если они равны. В цикле принимаем предложение и проверяем его на количество заглавных букв с помощью функции *check_sentence.* Если предложение подходит, то увеличиваем переменную *sentence_ add* на 1 и добавляем предложение в *text,* иначе увеличиваем переменную *sentence_delete* на 1. Не забываем проверять, не превысило ли количество добавленных предложений*(sentence_add)* выделенный размер *size*, если нужно — выделяем еще память и меняем с помощью функции *realloc*. Далее с помощью цикла *for* печатаем на экран корректный текст по предложениям, а также очищаем их из памяти. Затем очищаем массив указателей *text*. Печатаем количество предложений «до» и «после» изменения (количество на 1 меньше, так как конечное предложение Dragon flew away! В подсчет не идет).

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	vejd.vejd? GGvnjek;bhGbhJ.Dragon flew away!	vejd. vejd? Dragon flew away! Количество предложений до 4 и количество предложений после 2	Верный ответ
2.	Dragon flew away!	Dragon flew away! Количество предложений до 0 и количество предложений после 0	Верный ответ

Выводы.

Была изучена работа с указателями, строками, двумерными массивами и динамичной памятью в языке С. Разработана программа, считывающая текст, удаляющая вопросительные предложения и пробельные символы между предложениями, считающая количество предложений в тексте до и после выполнения программы, выводящая полученные предложения и значения на экран. Программа динамически выделяет память для хранения текста и предложений.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Valeeva_Alina_lb3.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define TEXT_SIZE 100

// функция для получения предложения в массив предложений
char* readSentence() {
    int size = TEXT_SIZE;
    int sentence_len = 0;
    char* sentence = malloc(size * sizeof(char));
    int symbol = getchar();
    if (symbol == '\n');
        // работа с табуляцией
    else {
        if (symbol != ' ' && symbol != '\t'){
            sentence[sentence_len++] = symbol;
        }
    }
    symbol = getchar();
    if (symbol != ' ' && symbol != '\t') {
        sentence[sentence_len++] = symbol;
    }
    do{
        symbol = getchar();
        sentence[sentence_len++] = symbol;
        if (sentence_len == size) {
            size += TEXT_SIZE;
            sentence = realloc(sentence, size);
        }
    }while (symbol != '.' && symbol != ';' && symbol != '?' &&
symbol != '!');
    sentence[sentence_len] = '\0';
    return sentence;
}
// проверка предложения на количество заглавных букв
int check_sentence(char *s){
    int count = 0;
    for(int i=0; i<strlen(s); i++){
        if(isupper(s[i])){
            count += 1;
        }
    }
    if(count > 1){
        return 0; // неподходящее
    }
    return 1; // подходящее
}

int main(){
    int size = TEXT_SIZE;
    char** text = malloc(size*sizeof(char*));
    char* end_sent = "Dragon flew away!";
```

```
        int sentence_add = 0, sentence_delete = 0; // sentence_add -
количество подходящих, sentence_delete - неподходящих
        char* sentence;
        sentence = readSentence();
        if (check_sentence(sentence)) { // если предложение подходит -
добавляем
            text[sentence_add++] = sentence;
        }
        else{
            sentence_delete += 1;
        }
        while (strcmp(sentence, end_sent) != 0) {
            sentence = readSentence();
            if (check_sentence(sentence)) { // если предложение
подходит - добавляем
                text[sentence_add++] = sentence;
            }
            else{
                sentence_delete += 1;
            }
            if (sentence_add == size) {
                size += TEXT_SIZE; // если память закончилась,
добавляем
                text = realloc(text, size * sizeof(char*));
            }
        }
        for (int i = 0; i < sentence_add; i++){
            puts(text[i]);
            free(text[i]);
        }
        free(text);

        printf("Количество предложений до %d и количество предложений
после %d\n", sentence_add + sentence_delete - 1, sentence_add - 1);
        return 0;
    }
```