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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Программирование»

Тема: Использование указателей в языке Си

Студент гр. 1304 _____ Маркуш А.Е.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2021

Цель работы.

Исследование использования указателей языка Си.

Задание.

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).

Основные теоретические положения.

В данной лабораторной работе использовались такие библиотеки, как *stdio.h*, *stdlib.h*, *string.h*.

Выполнение работы.

В функции *main()* объявляется переменная указатель на указатель типа *char*, которая имеет имя *text*. Она будет являться динамическим массивом строк, где каждая строка – это предложение. Так же объявляется переменная *text_size* типа *int,* и ей присваивается значение, возвращаемое функцией *get_text(char ***text).* Далее объявляется переменная *int new_size,* в которую передаётся значение, возвращаемое функцией *del_sentences(char ***text, int text_size)*. Далее вызываются функции *print_text(char **text, int pr_size, int*

new_size) и *free_rext(char ***text, int new_size)*. После этого программа заканчивает работу возвращая 0.

get_text() принимает адрес переменной *text*, для дальнейшего её изменения внутри функции. Далее объявляется переменная *int s_number = 0*, которая будет счётчиком предложений. Далее следует цикл *while*, который оканчивается, если последняя строка совпала с терминальной строкой. Внутри этого цикла динамически выделяется память под указатели на строки. Затем объявляется переменная *int c_number*, которая является счётчиком символов в строке. Функция *scanf(“ ”)* вызвана для игнорирования табуляции. Внутри этого цикла *while* вложен ещё один цикл *while*, в котором реализуется выделение памяти под символы и ввод символов с помощью функции *scanf()*. Память выделяется в обоих циклах по необходимости с помощью стандартных функций *realloc* и *malloc*. Вложенный цикл заканчивается, если последним символом были “!”, “?”, “.” или “;”. Затем функция возвращает значение *s_nymber*.

del_sentences() принимает адрес переменной *text*, для дальнейшего её изменения внутри функции и количество строк в тексте. В начале создаётся переменная указатель на указатель на *char*, в которая будет содержаться отредактированный текст. Затем объявляется переменная *new_size = 0.* Она будет счётчиком предложений в новом тексте. Далее идут два цикл *for,* одни из которых вложен в другой. Первый перебирает строки, а второй символы в этих строках. Внутри внешнего цикла объявляем переменную *int count = 0.* После чего идёт внутренний цикл. Если код символа больше 64 и меньше 91, то переменная *count* увеличивается на 1. Если после проверки всей строки *count* больше 1, то память выделенная под эту строку очищается, тем самым предложение удаляется. В противном случае выделяется память для новой строки в *n_text* и в *n_text[size]* передаётся строка, которая находится в *(*text)[i]*, где *i* счётчик внешнего цикла *for.* После проверки всех строк мы освобождаем память **text*, в которой находятся указатели на строки старого текста, а затем передаём в **text* значение *n_text.* В конце функция возвращает *n_size*.

print_text() принимает текст и его размер и с помощью цикла *for* и

функции *printf()* выводит на экран текст. Функция ничего не возвращает.

free_text() принимает адрес переменной с тестом и сначала освобождает память, выделенную под символы в каждой строке с помощью цикла *for* и функции *free*, а затем освобождает память выделенную под строки также с помощью функции *free*.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Look; After? Why? Dragon flew away!	Look; After? Why? Dragon flew away! Количество предложений до 3 и количество предложений после 3	Ответ верный
2.	Dragon flew away!	Dragon flew away! Количество предложений до 0 и количество предложений после 0	Ответ верный
3.	NO! Hi? Programm. square; Dragon flew away!	Hi? Programm. square; Dragon flew away! Количество предложений до 4 и количество предложений после 3	Ответ верный
4.	Dragon flew away! No! YES?	Dragon flew away! Количество предложений до 0 и количество предложений после 0	Ответ верный

Выводы.

Было изучено использование указателей в языке СИ, а так же реализована программа по считыванию текста в динамический массив строк, его обработке и выводу на экран.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int get_text(char*** text){
    int s_number = 0;
    *text = NULL;
    do{
        *text = realloc(*text, sizeof(char*)*(s_number+1));
        int c_number = 0;
        (*text)[s_number] = NULL;
        (*text)[s_number] = malloc(sizeof(char)*(c_number+1));
        scanf(" ");
        do{
            scanf("%c", &(*text)[s_number][c_number]);
            c_number++;
            (*text)[s_number] = realloc((*text)[s_number],
sizeof(char)*(c_number+1));
        }while(((*text)[s_number][c_number-1] != '!') &&
((*text)[s_number][c_number-1] != '?') && ((*text)[s_number][c_number-
1] != '.') && ((*text)[s_number][c_number-1] != ';'));

        (*text)[s_number][c_number] = '\0';
        s_number++;
    }while(strcmp((*text)[s_number-1], "Dragon flew away!") != 0);

    return s_number;
}

int del_sentences(char*** text, int text_size){

    char** n_text = NULL;
    int n_size = 0;

    for(int i = 0; i < text_size; i++){
        int count = 0;
        for(int j = 0; j < strlen((*text)[i]); j++){
            if((*text)[i][j] > 64 && (*text)[i][j] < 91){
                count++;
            }
        }
        if(count > 1){
            free((*text)[i]);
        }
        else{
            n_text = realloc(n_text, sizeof(char*)*(n_size+1));
            n_text[n_size] = (*text)[i];
            n_size++;
        }
    }
    free(*text);
    *text = n_text;
    return n_size;

}

void print_text(char **text, int pr_size, int new_size){
    for(int i = 0; i < new_size; i++){
```

```
        printf("%s\n", text[i]);
    }
    printf("Количество предложений до %d и количество предложений
после %d", pr_size-1, new_size-1);
}

void free_text(char ***text, int size){

    for(int i = 0; i < size; i++){
        free((*text)[i]);
    }
    free((*text));

}


int main(){
    char **text;
    int text_size = get_text(&text);
    int new_size = del_sentences(&text, text_size);

    print_text(text, text_size, new_size);
    free_text(&text, new_size);

    return 0;
}
```