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 _____ Поршнев Р.А.

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 *sentences*. Она будет являться динамическим массивом строк, где каждая строка – это предложение. Переменной *sentences* присваивается значение функции *read_text()*.

В функции *read_text()* объявляется переменная *c* типа *char*. В неё будет записываться каждый новый вводимый символ. Далее объявляется переменная *text*, которая будет являться указателем на указатель типа *char*. Затем следует инициализация переменной *sentences* значением 0, а также объявляется переменная *symbols* типа *int*. Переменная *sentences* нужна для подсчёта количества предложений, а переменная *symbols* нужна для подсчёта количества символов в соответствующей строке. Переменной *text* присваивается нулевой указатель.

Далее следует цикл *while* со следующим условием окончания: последняя строка равна терминальной. Затем динамически выделяется память под указатели на строки с помощью *realloc*. Память будет выделяться по мере необходимости. Переменной *symbols* присваивается 0. С помощью функции *scanf(" ")* игнорируется табуляция.

Затем следует цикл *while* со следующим условием окончания: вводимый символ равен либо “.”, либо “;”, либо “?”, либо “!”. Далее происходит считывание вводимого символа. После динамически выделяется память под символы. Память будет выделяться по мере надобности для каждого вводимого символа. Вводимый символ заносится массив *text*.

После завершения предыдущего цикла выделяется память под последний символ, который будет являться символом конца строки. Данный символ заносится в массив *text*.

В функции *main()* переменная *sentences_count* типа *int* инициализируется значением функции *count(char **sentences)*. Данная функция подсчитывает количество предложений.

В качестве аргумента функция *count(char **sentences)* принимает введённый текст, разделённый на предложения. Переменная *i* типа *int* инициализируется значением 0. Далее следует цикл *while* со следующим условием окончания: данное предложение равно терминальному предложению. В цикле ведётся подсчёт количества предложений. Функция возвращает количество предложений в тексте, без учёта терминального.

В функции *main()* переменной *sentences* присваивается значение функции *filter(char **sentences, int sentences_count)*.

В функции *filter(char **sentences, int sentences_count)* объявляются переменные *i* и *j* типа *int*, которые будут счётчиками в циклах, переменная *i* инициализируется значением 0. Объявляется переменная *check* типа *int*, основываясь на значении которой будет определяться: удалить предложение или нет. Далее следует обход текста посимвольно. Если код символа больше 64 и меньше 91, то переменную *check* увеличиваем на 1. Таким образом происходит проверка: является ли буква заглавной. Если значение *check* больше 1, то данное предложение удаляется с помощью функции *free*. Далее происходит смещение массива строк на 1 влево. После обхода текста функции возвращается массив строк, в каждой из которых не более двух заглавных букв.

В функции *main()* переменная *new_sentences_count* типа *int* инициализируется значением функции *count(char **sentences)*. Далее вызывается функция *printtext(char **sentences, int n).*

В функции *printtext(char **sentences, int n)* объявляется переменная *i* типа *int*, которая будет счётчиком в цикле. В цикле происходит вывод предложений, в которых не более двух заглавных букв. Функция ничего не возвращает.

Далее в функции *main()* происходит вывод количества предложений до и после удаления предложений с двумя и более заглавными буквами. Так же происходит освобождение памяти двумерного динамического массива с помощью функции *free_text(char** sentences, int n)*.

В функции *free_text(char** sentences, int n)* объявляется переменная *i* типа *int*. Далее происходит обход цикла, в котором освобождается память под символы строк. После завершения цикла очищается память под указатели на строки двумерного массива.

Функция *main()* оканчивается тем, что ей возвращается значение 0.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	sweden; lolo. c++? Dragon flew away!	sweden; lolo Dragon flew away! Количество предложений до 3 и количество предложений после 3	Ответ правильный
2.	Dragon flew away!	Dragon flew away! Количество предложений до 0 и количество предложений после 0	Ответ правильный
3.	NEstle; Codeblocks. Programming on C? Dragon flew away!	Codeblocks. Dragon flew away! Количество предложений до 3 и количество предложений после 1	Ответ правильный
4.	Dragon flew away! NEstle; Codeblocks. Programming on C? Dragon flew away!	Dragon flew away! Количество предложений до 0 и количество предложений после 0	Ответ правильный

Выводы.

Я исследовал использование указателей в языке Си.

Разработана программа, считывающая текст и записывающая его в динамический массив строк. Также данная программа производит обработку предложений по заданному условию и выводит их на экран.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.c

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


char** read_text(){

    char c;
    char **text;
    int sentences = 0, symbols;
    text = NULL;
    do {
        text = realloc(text, (sentences + 1) * sizeof(char*));
        text[sentences] = NULL;
        symbols = 0;
        scanf(" ");
        do{
            scanf("%c", &c);
        text[sentences] = realloc(text[sentences], (symbols +  1)
* sizeof(char));
            text[sentences][symbols] = c;
            symbols += 1;
         } while ((c != '.') && (c != ';') && (c != '?') && (c !=
'!'));
          text[sentences] = realloc(text[sentences], (symbols + 1)
* sizeof(char));
        text[sentences][symbols] = '\0';
        sentences += 1;
     } while (strcmp(text[sentences - 1], "Dragon flew away!") !=
0);
    return text;
}


char** filter(char **sentences, int sentences_count){

    int i = 0, check = 0, j;
    while (i < sentences_count){
        check = 0;
        for (j = 0; j < strlen(sentences[i]); j++)
            if ((sentences[i][j] > 64) && (sentences[i][j] < 91))
                check += 1;
        if (check > 1){
            free(sentences[i]);
            for(j = i; j < sentences_count - 1; j++)
                sentences[j] = sentences[j + 1];
            sentences_count--;
        }
        else
            i++;
    }
    return sentences;
}
```

```
int count(char **sentences){

    int i = 0;
    while (strcmp(sentences[i], "Dragon flew away!") != 0){
        i = i + 1;
    }
    return i;
}


void printtext(char **sentences, int n){

    int i;
    for(i = 0; i < n; i++)
        printf("%s\n", sentences[i]);
}


void free_text(char** sentences,  int n){

    int i;
    for(i = 0; i < n; i++)
        free(sentences[i]);
    free(sentences);
}


int main(){

    int i;
    char **sentences = read_text();
    int sentences_count = count(sentences);
    sentences = filter(sentences, sentences_count + 1);
    int new_sentences_count = count(sentences);
    printtext(sentences, new_sentences_count + 1);
    printf("Количество предложений до %d и количество предложений
после %d\n", sentences_count, new_sentences_count);
    free_text(sentences, new_sentences_count + 1);
    return 0;
}
```