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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Программирование»
Тема “Использование указателей”

Студентка гр. 1384 _____ Логинова А. Ю.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2021

Цель работы.

Научиться работать с указателями и символьными массивами в C, написать программу, обрабатывающую текст динамически.

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 (\t, ' ') в начале предложения должна быть удалена.
- Все предложения, в которых есть число 555, 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.

На вход программе подается текст, который считывается посимвольно, переменной *symbol* присваивается значение символа. Для работы с текстом был реализован двумерный массив указателей *array*, элементами которого являются указатели на массивы символов. Для каждого элемента выделяется память из кучи (*heap*) с помощью функций *malloc()* и *realloc().*

Если первый символ в предложении это *whitespace* (‘\t’, ‘ ‘, ‘\n’), программа не записывает этот символ в массив. Индексами массива *array* являются переменные *current_sentence* и *current_char*, благодаря которым можно записывать значения *symbol* в массив. Далее создается временная переменная *temp,* равная *array* типа *char*,* в конец которой добавляется нуль-терминатор (‘\0’) и *temp* сравнивается с терминальным предложением. Если *temp* является терминальным предложением, программа выходит из цикла и перестает считывать текст.

Если *temp* не является терминальным предложением, программа перезапишет на месте нуль-терминатора *symbol* и будет проверять дальше. Цикл завершается, когда программа встречает разделитель (одно из значений: ‘.’, ’;’, ‘?’), прибавляет в конец нуль-терминатор, обнуляет *current_char.* Таким образом заполняется массив предложениями, разделенными по символам-разделителям.

Когда массив заполнится, программа должна вывести только те предложения, в которых нет ‘555’ как отдельного слова. В переменных *amount* и *upgraded_amount* лежат количество предложений изначально (без терминального предложения) и количество оставшихся предложений (без терминального предложения). Программа выводит их соответственно.

Код программы см. в приложении А.

Тестирование.

№ п/п	Входные данные	Выходные данные	Комментарии
1	Nulla facilisi. Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra, per inceptos himenaeos. 40 Nu555lla rutrum feugiat felis a pharetra. Sed finibus magna et mauris elementum tempus? Integer at quam et erat iaculis iaculis hendrerit a te4llus? Donec at nunc ac mauris suscipit venenatis. Sed finibus magna et mauris elementum tempus? Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi co7ndimentum 555 ex justo, nec pharetra mauris vestibulum a. Suspendisse quis mi neque7. 1 Vivamus eu nibh rhoncus, da456pibus ex non, sodales mi. Donec accumsan convallis ipsum vitae lacinia. Donec accumsan convallis ipsum vitae lacinia. Fusce finibus sapien magna, quis scelerisque ex sodales	Nulla facilisi. Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra, per inceptos himenaeos. 40 Nu555lla rutrum feugiat felis a pharetra. Sed finibus magna et mauris elementum tempus? Integer at quam et erat iaculis iaculis hendrerit a te4llus? Donec at nunc ac mauris suscipit venenatis. Sed finibus magna et mauris elementum tempus? Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse quis mi neque7. 1 Vivamus eu nibh rhoncus, da456pibus ex non, sodales mi. Donec accumsan convallis ipsum vitae lacinia. Donec accumsan convallis ipsum vitae lacinia.	Программа работает корректно.

	tristique. Nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Dragon flew away!	Fusce finibus sapien magna, quis scelerisque ex sodales tristique. Nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Dragon flew away! Количество предложений до 16 и количество предложений после 15	

Выводы.

Были изучены возможности указателей в C, символьных массивов, динамического выделения памяти, функций работы со строками.

ПРИЛОЖЕНИЕ А.

Исходный код программы.

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define DEFAULT_ARR_SIZE 1
#define DEFAULT_STRING_SIZE 1


int clear_555(char* str)
{
    if(strstr(str, " 555 ") || strstr(str, " 555.")
    || strstr(str, " 555;") || strstr(str, " 555?")
    || (strstr(str, "555 ") == str))
    {
        return 1;
    }
    return 0;
}

int main(){
    int current_arr_size = DEFAULT_ARR_SIZE, current_string_size =
DEFAULT_STRING_SIZE;
    char **array = (char**)malloc(current_arr_size * sizeof(char*));
    int current_sentence = -1, current_char = 0;
    char symbol, *temp;

    do
    {
        // Проверка на наличие памяти
        if (current_sentence >= current_arr_size)
            {
                current_arr_size *= 2;
                array = (char**)realloc(array, current_arr_size *
sizeof(char*));
            }


        current_sentence++;
        array[current_sentence] = (char*)malloc(current_string_size *
sizeof(char));

        do
        {
            // Проверка на наличие памяти
            if (current_char >= current_string_size)
            {
                current_string_size *= 2;
                array[current_sentence] =
(char*)realloc(array[current_sentence], current_string_size *
sizeof(char));
            }
```

```
            // Считывание символа
            symbol = getc(stdin);


            // Убирает whitespaces в начале предложения
            if (!(current_char == 0 && (symbol == ' ' || symbol == '\t'
|| symbol == '\n')))
            {
            array[current_sentence][current_char] = symbol;
            current_char++;
            }


            // Проверка на терминальное предложение
            temp = array[current_sentence];
            temp[current_char] = '\0';
            if(strcmp("Dragon flew away!", temp) == 0)
            {
                break;
            }


        } while(symbol != '.' && symbol != '?' && symbol != ';');


    array[current_sentence][current_char] = '\0';
    current_char = 0;


    } while(strcmp("Dragon flew away!", array[current_sentence]) != 0);

    int amount = current_sentence, upgraded_amount = -1;

    int i;

    for(i = 0; i < current_sentence + 1; ++i)
    {

        if (clear_555(array[i]) == 0)
        {
            upgraded_amount++;
            printf("%s\n", array[i]);

        }


    }

    printf("Количество предложений до %d и количество предложений после
%d\n", amount, upgraded_amount);

    return 0;
}
```