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

Студентка гр. 0382 _____ Здобнова К.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучить указатели. Научиться динамически выделать память, а также ее освобождать.

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.

С помощью *malloc* выделяется память для двумерного массива *output_text*, куда записывают указатели на предложения. Предложения считываются с помощью функции *get_sentence()*. Входные данные считываются посимвольно с помощью функции стандартной библиотеки *getchar()*. В *get_sentence()* помимо

считывания выполняется ряд преобразований: в окончательном виде из предложения убираются знак табуляции, пробел перед началом, а также двойные пробелы. Предложения записываются в массив sentence, паять для которого выделяется с помощью *malloc*, при переполнении (изначально *BUFFER* = 100) выделяется дополнительно с помощью *realloc*. Количество символов в предложении записывается в *len_of_sentence*).

Функция *string_checker(char * sentence)* проверяет предложения на корректность. Если в предложении больше одной заглавной буквы, то оно не записывается в массив *output_text*.

Терминальное состояние хранится в переменной *end_of_text*, с помощью функции *strcmp()* проверяются предложения на маркерное предложение, если результат ее выполнения 0, то программа выходит из цикла обработки входных данных.

В переменной *size_of_otput_text* записывается число всех предложений, а в *counter_rigth_sentences* количество правильных.

С помощью функции *free()*, освобождается память, выделенная для двумерного массива.

На экран выводится элементы массива *output_text*, терминальное предложение и фраза "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 самой фразы).

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.	Fasas trh5gdhk sdjhhd, sdfsfd - sdfdsf. asdfG dsfdH sdfs. Rvsskjdh asidua asaf 55 dfsdf, dsfsfdhs ttyd. Rteyyss GGG sfjdfhd Dfgshsefj. Yep; Sdfsdfs sdjfsh eyrwt lsfdk. NO. Fdghgds, sdsd. Dragon flew away!	Fasas trh5gdhk sdjhhd, sdfsfd - sdfdsf. Rvsskjdh asidua asaf 55 dfsdf, dsfsfdhs ttyd. Yep; Sdfsdfs sdjfsh eyrwt lsfdk. Fdghgds, sdsd. Dragon flew away! Количество предложений до 8 и количество предложений после 5	Программа выводит правильный ответ.
2.	Fsgdfsh dfsdfh sfsdf. ADdsda asda asdsd? asd sfdsf? sjfgdhd. Dragon flew away! Fsgdfsh dfsdfh sfsdf. ADdsda asda asdsd? asd sfdsf? sjfgdhd. Dragon flew away!	Fsgdfsh dfsdfh sfsdf. asd sfdsf? sjfgdhd. Dragon flew away! Количество предложений до 4 и количество предложений после 3	Программа выводит правильный ответ.
3.	Are ssdfh sdfs 4533 fdfd. sdfhs sdfHHad sdfs? fsfs; Casas fsh jskdfk jsfdsjdsS; R? fsdfj sfdshj Dfsfd. Some tjfu vsvdvfv687dvd sdfsdj. Dragon flew away!	Are ssdfh sdfs 4533 fdfd. fsfs; R? fsdfj sfdshj Dfsfd. Some tjfu vsvdvfv687dvd sdfsdj. Dragon flew away! Количество предложений до 7 и количество предложений после 5	Программа выводит правильный ответ.

Выводы.

Был изучен алгоритм динамического выделения и освобождения памяти. Была изучена библиотека работы со строками, а так же работа с указателями.

Разработана программа, считывающая с клавиатуры текст и преобразующая его в двумерный массив.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: *lb3.c*

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

#define BUFFER 100

char* get_sentence(){
    int symbol;
    char* sentence = (char*) malloc(BUFFER * sizeof(char));
    int len_of_sentence = 0;
    int buf = BUFFER;
    symbol = getchar();
    if ((len_of_sentence != 0 && symbol == ' ')){
            sentence[len_of_sentence] = symbol;
            len_of_sentence++;
    }else if((len_of_sentence == 0 && symbol != ' ')){
            sentence[len_of_sentence] = symbol;
            len_of_sentence++;
    }
    while (1){
            symbol = getchar();
            if (len_of_sentence != 0)
                if (symbol == ' ' && sentence[len_of_sentence - 1] ==
' ')
                    continue;
            sentence[len_of_sentence] = symbol;
            len_of_sentence++;
            if (symbol == '.' || symbol == ';' || symbol == '?' ||
symbol == '!')
                break;
            if (len_of_sentence == buf){
                buf += BUFFER;
                sentence = (char*) realloc(sentence, buf);
            }
    }
    sentence[len_of_sentence] = '\0';
    return sentence;
}

int string_checker(char * sentence){
    int num_of_Up = 0;
    for (int i = 0; i < strlen(sentence); i++){
        if (sentence[i] >= 'A' && sentence[i] <= 'Z')
            num_of_Up++;
        if (num_of_Up > 1)
            return 0;
    }
    return 1;
}

int main()
{
    char* end_of_text = "Dragon flew away!\0";
```

```
    char* sentence = " ";
    char** output_text = (char**) malloc(BUFFER * sizeof(char*));
    int size_of_otput_text = 0, buf = BUFFER, counter_all_sentences =
0, counter_rigth_sentences = 0;
    while(1){
        sentence = get_sentence();
        counter_all_sentences++;
        if (strcmp(sentence, end_of_text) == 0)
            break;
        if(string_checker(sentence)){
            output_text[size_of_otput_text] = sentence;
            size_of_otput_text++;
            counter_rigth_sentences++;
        }
        if (size_of_otput_text == buf){
            buf += BUFFER;
            output_text = (char**) realloc(output_text,
buf*sizeof(char*));
        }
    }
    for(int i = 0; i < size_of_otput_text; i++){
        puts(output_text[i]);
        free(output_text[i]);
    }
    printf("Dragon flew away!\n");
    printf("Количество предложений до %d и количество предложений
после %d", counter_all_sentences - 1, counter_rigth_sentences);
    return 0;
    free(output_text);
}
```