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. 9304 _____ Каменская Е.К.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2019

Цель работы.

Реализовать алгоритмы считывания, обработки и вывода текста при помощи указателей и динамических массивов.

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).

* Порядок предложений не должен меняться

* Статически выделять память под текст нельзя

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

Основные теоретические положения.

Указатель – это переменная, содержащая адрес другой переменной.

Синтаксис объявления указателя:

<тип_переменной_на_которую_ссылается_указатель>* <название переменной>;

Для работы с динамической памятью используются следующие функции:

malloc (void malloc (size_t size))* - выделяет блок из size байт и возвращает указатель на начало этого блока

calloc (void calloc (size_t num, size_t size))* - выделяет блок для num элементов, каждый из которых занимает size байт и инициализирует все биты выделеного блока нулями

realloc (void realloc (void* ptr, size_t size))* - изменяет размер ранее выделенной области памяти на которую ссылается указатель ptr. Возвращает указатель на область памяти, измененного размера.

free (void free (void ptr))* - высвобождает выделенную ранее память.

Выполнение работы.

В программе реализованы следующие функции:

- *int read_til_terminal(char*** totext, char* terminal)* – считывает текст по предложению за итерацию, вызывая функцию *get_sentence*, и сравнивает полученное предложение с терминальным с помощью функции *strcmp*. После того, как встретит терминальное, возвращает количество считанных предложений (минус последнее). На вход получает адрес указателя на уже существующий массив указателей и терминальное предложение в виде массива char.
- *char* get_sentence()* – создает массив типа char и считывает в него из потока *stdin* посимвольно предложение, пока не встретит один из символов окончания предложения. Возвращает указатель на созданный массив.
- *int del_by_feature(char** text, int n)* – получает на вход указатель на первый элемент массива массивов *char* (указатель на указатель на первое предложение) и количество элементов массива предложений и итерируется по данному массиву. Находя предложение с более, чем одной заглавной буквой,

удаляет его, используя функцию *free()* и сдвигает все указатели в массиве на один влево. Возвращает новое количество предложений.

- *int main()* – основная функция, в которой создается двумерный массив для хранения предложений *text*. Переменные n и m инициализируются результатами работы функций *read_til_terminal* и *del_by_feature* соответственно. С помощью цикла *for* и функции *puts* происходит вывод отформатированного текста на экран. Последним выводится терминальное предложение, записанное в *text[n]*, не учитываемое в процессе форматирования. После вывода все предложения удаляются функцией *free()*, как и массив *text* в самом конце.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	All in the golden afternoon. Full leisurely we glide; For both our oars, with little skill? By little arms are plied. Dragon flew away!	All in the golden afternoon. Full leisurely we glide; For both our oars, with little skill? By little arms are plied. Dragon flew away! Количество предложений до 4 и количество предложений после 4
2.	While little hands make vain pretence; Our wanderings to guide. Ah, cruel Three; In such an hour, Beneath such dreamy weather; To beg a tale of breath too weak. Dragon flew away!	While little hands make vain pretence; Our wanderings to guide. To beg a tale of breath too weak. Dragon flew away! Количество предложений до 5 и количество предложений после 3

Выводы.

В ходе работы были изучены функции библиотеки *stdlib* для работы с динамической памятью и принципы использования указателей. Разработана

программа, выполняющая посимвольное считывание текста из потока *stdin* в динамический двумерный массив, форматирующая его согласно заданию и выводящая результат на экран. Для считывания используется цикл с постусловием *do while* и функция *getc()*. Для выделения динамической памяти используются функции *malloc()* и *realloc()*, для ее освобождения – *free()*. Во избежание утечки памяти и потери считанного текста, в функцию считывания *read_til_terminal()* передается указатель на указатель на двумерный массив, благодаря чему *realloc* меняет указатель *text*, созданный в основной функции.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: var5.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define TERM "Dragon flew away!"

int read_til_terminal(char*** totext, char* terminal);

char* get_sentence();

int del_by_feature(char** text, int n);

int main(){
    char** text = malloc(1*sizeof(char));
    int n = read_til_terminal(&text, TERM);
    int m = del_by_feature(text, n);
    for(int i=0; i<m; i++){
        puts(text[i]);
        free(text[i]);
    }
    puts(text[n]);
    free(text[n]);
    free(text);
     printf("Количество предложений до %d и количество предложений
после %d", n, m);
    return 0;
}

int read_til_terminal(char*** totext, char* terminal){
    int n = 0;
    do{
        *totext = realloc(*totext, (++n)*sizeof(char*));
        *(*totext+n-1) = get_sentence();

    }while(strcmp(*(*totext+n-1), terminal));
    //free(*(*totext+n-1)); //terminal sentence deletion
    return n-1;
}

char* get_sentence(){
    char* newsntc = malloc(1*sizeof(char));
    char c;
    int i = 0;
     while((c = getc(stdin)) && (c != '.') && (c != '?') && (c !=
';') && (c != '!')){
        newsntc[i] = c;
        i++;
        newsntc = realloc(newsntc, (i+1)*sizeof(char));
    }
    newsntc = realloc(newsntc, (i+2)*sizeof(char));
    newsntc[i++] = c;
    newsntc[i] = '\0';
```

```
    c = getc(stdin);
    return newsntc;
}

int del_by_feature(char** text, int n){
    int count = 0;
    int i = 0;
    while(i<n){
        count = 0;
        for(int j = 0; j<strlen(text[i]); j++){
            if(isupper(text[i][j]))
                count++;
        }
        if(count > 1){
            free(text[i]);
            for(int u = i+1; u<n; u++)
                text[u-1] = text[u];
            n--;
        }
        else{
            i++;
        }
    }
    return n;
}
```