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

Студент гр. 0382 _____ Мукатанов А.В.
Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение работы с указателями и динамической памятью в языке 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).

* 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.

Функции stdlib.h:

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

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.

Циклы:

If {...} else {…}

for(…){…}

while (…){…}

Выполнение работы.

Для начала используются глобальные переменные SIZE и ADD_MORE они отвечают за количество выделенной памяти.

Для считывания символов используется функция char* reading().

Для сравнивания строк используется функция int compare(char* i_str, c_str)

Для удаления строки , в которой присутствует более одного символа в верхнем регистре , используется функция char* delete(char* b_str)

Для проверки того - является ли первый элемент символьного массива пробелом, табуляцией или новой строкой. В цикле так же находится другой цикл for смещающий каждый элемент массива на 1 влево если какой либо отступ в начале предложения был найден. Используется функция char* fixing(char* a_str).

Функция int main():

В данной функции осуществляется цикл while ,который отвечает за выделение памяти. И цикла for , который является счетчиком предложений.

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.	Aenean sem ligula, laoreet ac sodales a, congue euismod neque; Aenean magna massa, scelerisque quis sagittis at, pharetra a lectus? Integer lAoreet5 vene45natis	Aenean sem ligula, laoreet ac sodales a, congue euismod neque; Aenean magna massa, scelerisque quis sagittis at, pharetra a lectus?	Программа работает правильно

	ullamcorper? Nu555llam auctor vehicula dui, quis lo-bortis nibh. Vivamus sit amet vi-verra arcu, sed ultricies nulla. Dragon flew away!	Nu555llam auctor vehicula dui, quis lobortis nibh. Vivamus sit amet viverra arcu, sed ultricies nulla. Dragon flew away! Количество предложений до 5 и количество предложений после 4	
2.	Maecenas 555 posuere velit efficitur, egestas nunc quis, dictum purus? Nam 7elementum id enim eu congue; Morbi co7ndimentum 555 ex justo, nec pharetra mauris vestibulum a. Ut auctor augue vel tincidunt tincidunt 555. 1 Vivamus eu nibh rhoncus, da456pibus ex non, sodales mi. Cras eget felis nibh? Aliquam at ultricies nisl, sed pretium nulla; Fusce finibus sapien magna, quis scelerisque ex sodales tristique. Aenean sem ligula, laoreet ac sodales a, congue euismod neque; Aenean magna massa, scelerisque quis sagittis at, pharetra a lectus? Integer lAoreet5 vene45natis ullamcorper? Nu555llam auctor vehicula dui, quis lo-bortis nibh. Vivamus sit amet vi-verra arcu, sed ultricies nulla. Dragon flew away!	Maecenas 555 posuere velit efficitur, egestas nunc quis, dictum purus? Nam 7elementum id enim eu congue; Morbi co7ndimentum 555 ex justo, nec pharetra mauris vestibulum a. Ut auctor augue vel tincidunt tincidunt 555. 1 Vivamus eu nibh rhoncus, da456pibus ex non, sodales mi. Cras eget felis nibh? Aliquam at ultricies nisl, sed pretium nulla; Fusce finibus sapien magna, quis scelerisque ex sodales tristique. Aenean sem ligula, laoreet ac sodales a, congue euismod neque; Aenean magna massa, scelerisque quis sagittis at, pharetra a lectus? Nu555llam auctor	Программа работает верно

		vehicula dui, quis lobortis nibh. Vivamus sit amet viverra arcu, sed ultricies nulla. Dragon flew away! Количество предложений до 13 и количество предложений после 12	

Выводы.

В ходе работы была изучена работа с динамической памятью и указателям

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab_3.c

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

#define BUFFER 50
#define ADD_MORE 50

char* delete(char* b_str){
    int count = 0;
    for(int i = 0; i < strlen(b_str); i++){
        if(isupper(b_str[i])){
            count++;
        }
    }
    if(count > 1){
        b_str = "\0";
    }
    return b_str;
}

char* reading(){
    int len = 0, size = BUFFER, a;
    char *text = malloc(size*sizeof(char));
    while(1){
        a = getchar();
        text[len++] = a;
```

```
        if(len == size){
            size += ADD_MORE;
            text = realloc(text, size);
        }
        if(a == '.' || a == ';' || a == '?' || a == '!')
            break;
    }
    text[len]='\0';
    return text;
}

char* fixing(char* a_str){
    int a_len = strlen(a_str);
    while(a_str[0] == ' ' || a_str[0] == '\t' || a_str[0] == '\n'){
        for(int i = 0; i < a_len; i++){
            a_str[i] = a_str[i + 1];
        }
    }
    a_str = delete(a_str);
    return a_str;
}



int main(){
    char* default_str = "Dragon flew away!";
    char** arr = malloc(BUFFER*sizeof(char*));
    int arrlen = 0, counter = 0, i;
    char* str;
    int size_arr = BUFFER;
```

```
while(1){
    str = reading();
    str = fixing(str);
    arr[arrlen] = str;
    if(!strcmp(arr[arrlen++], default_str)){
        break;
    }
    if(arrlen == size_arr){
        size_arr += ADD_MORE;
        arr = realloc(arr, size_arr*sizeof(char*));
    }
}
for(i = 0; i <= arrlen;){
    if(!strcmp(arr[i], default_str)){
        i++;
        break;}
    if(!strcmp(arr[i], "\0")){
        for(counter = i; counter < arrlen; counter++){
            arr[counter] = arr[counter+1];}
    }
    else
        i++;

}
for (int j = 0; j < i; j++){
    puts(arr[j]);
    free(arr[j]);
}
printf("Количество предложений до %d и количество предложений после %d", arrlen-1, i-1);
```

```
    free(arr);
    return 0;
}
```