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 _____ Охотникова Г.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Научиться работать с текстом в языке C с помощью символьных 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, которые заканчиваются на '?' 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.

Указатель – это переменная, содержащая адрес другой переменной.

* – оператор разыменования.

& – оператор взятия адреса.

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.

*char *txt* –* одно предложение текста.

int size_arr – размер памяти, выделяемой для двумерного массива.

*char** arr_txt* – двумерный массив, в который будут записываться предложения текста.

len_arr – длина двумерного массива.

int n, m – счетчики количества предложений до преобразования и после соответственно.

char end* – переменная, в которую записано предложение, которым заканчивается ввод текста.

char readSentence()* – функция, с помощью которой посимвольно считывается каждое предложение с помощью *getchar()*. Так же в функции выделяется фиксированная память для предложения и дополнительная, если это необходимо, с помощью функций malloc и *realloc* соответственно.

int stop_input(char str1, char* str2)* – функция, которая принимает на вход предложение, которым заканчивается ввод текста и текущее считанное предложение. В данной функции происходит сравнение двух предложений, и, если они одинаковые, функция возвращает единицу.

char delete_tab(char *str)* – функция, которая принимает на вход текущее считанное предложение и удаляет в нем пробелы.

*int extra(char *str)* – функция, которая принимает на вход текущее считанное предложение и возвращает ноль, если оно заканчивается на вопросительный знак.

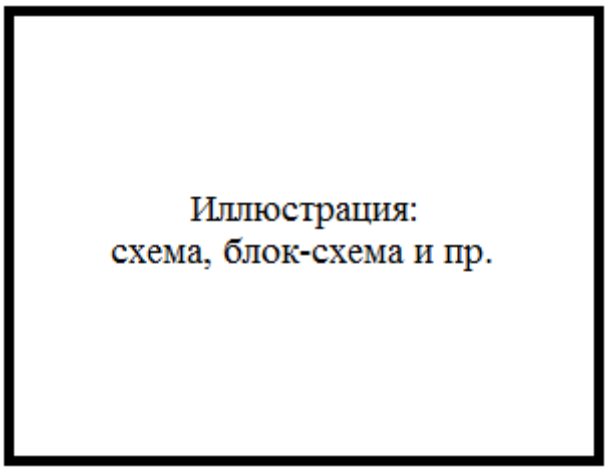

Рисунок 1 - Пример иллюстрации

В функции *main()* в цикле *while* сначала вызывается функция, считывающая предложение, затем функция, удаляющая пробелы. После этого к счетчику, который отвечает за количество предложений, добавляется единица. Затем, если предложение не заканчивается на вопросительный знак(функция *extra()* вернула единицу), к счетчику предложений после преобразования добавляется единица, а данное предложение записывается в двумерный массив текста. Затем выделяется дополнительная память с помощью функции *realloc,* если это необходимо. После происходит вызов функции на проверку конца ввода и, если функция *stop_input()* вернула единицу, происходит выход из цикла. В цикле *for* происходит вывод текста и освобождение памяти, которая была выделена на каждый одномерный

массив. Затем освобождается память, выделенная на работу с двумерным массивом.

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.	40 Nu555lla rutrum feugiat felis a pharetra. Integer at quam et erat iaculis iaculis hendrerit a te4llus? Morbi co7ndimentum 555 ex justo, nec pharetra mauris vestibulum a. Suspendisse quis mi neque7. Sed finibus magna et mauris elementum tempus? Sed finibus magna et mauris elementum tempus? Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra, per inceptos himenaeos. Donec at nunc ac mauris suscipit venenatis. Nam 7elementum id enim eu congue;	40 Nu555lla rutrum feugiat felis a pharetra. Morbi co7ndimentum 555 ex justo, nec pharetra mauris vestibulum a. Suspendisse quis mi neque7. Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra, per inceptos himenaeos. Donec at nunc ac mauris suscipit venenatis. Nam 7elementum id enim eu congue; Donec accumsan convallis ipsum vitae lacinia. Dragon flew away! Количество предложений до 10 и количество предложений после 7	Программа работает правильно.

	Donec accumsan convallis ipsum vitae lacinia. Dragon flew away!		
2.	Name miles eat answer made reserved first rapturous differed. Prevailed for satisfied admitting necessary sufficient ample order total service securing man removing secure out plate. Taste delay nature quit could service behind even means what remember. Favourite tore resources merit. Because terminated without rendered added quitting humoured find time attention part wishing rather comfort viewing held celebrated? Reached up eyes vicinity. Leave fond tall game one leaf learn such screened behind temper warrant nothing fortune john. Bred feel remember studied perpetual deficient. Dull savings horrible service respect objection excuse merit questions indulgence sweetness relied power. Rejoiced this improving.	Name miles eat answer made reserved first rapturous differed. Prevailed for satisfied admitting necessary sufficient ample order total service securing man removing secure out plate. Taste delay nature quit could service behind even means what remember. Favourite tore resources merit. Reached up eyes vicinity. Leave fond tall game one leaf learn such screened behind temper warrant nothing fortune john. Bred feel remember studied perpetual deficient. Dull savings horrible service respect objection excuse merit questions indulgence sweetness relied power. Rejoiced this improving. Dragon flew away! Количество предложений до 10 и количество предложений после 9	Программа работает правильно.

	Dragon flew away!		

Выводы.

Были изучены основные управляющие конструкции языка для работы с символьными массивами.

Разработана программа, выполняющая считывание с клавиатуры исходных данных в динамические массивы символов. Программа обрабатывает текст с помощью функций и выводит полученный результат.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define INIT_SIZE 50
#define DELTA 50
#define TRUE 1

char* readSentence() {
    int size = INIT_SIZE;
    int length = 0;
    char *text = malloc(size * sizeof(char));
    int c;

    while (TRUE) {
        c = getchar();
        text[length++] = c;
        if (length == size) {
            size += DELTA;
            text = realloc(text, size);
        }
        if (c == '.' || c == ';' || c == '?' || c == '!') {
            break;
        }
    }

    if (length > 0) {
        text[length] = '\0';
        return text;
    }
}

int stop_input(char* str1, char* str2) {
    int fact, i;
    for (i = 0; i < strlen(str2); i++) {
        if (str1[i] == str2[i]) {
            fact = 1;
        }
        else {
            fact = 0;
        }
    }
    return fact;
}

char* delete_tab(char *str) {
    int i = 0;
    int k = 0;
    while (str[i] == ' ' || str[i] == '\t' || str[i] == '\n') {
        int j;
```

```
        for (j = 0; j < strlen(str) - 1; j++) {
            str[j] = str[j + 1];
        }
        k++;
    }
    str[strlen(str) - k] = '\0';
    return str;
}



int extra(char *str) {
    int i = 0;
    int k = 1;
    for (i = 0; i < strlen(str); i++) {
        if (str[i] == '?') {
            k = 0;
        }
    }
    return k;
}

int main() {
    char *txt;
    int size_arr = INIT_SIZE;
    char** arr_txt = malloc(size_arr * sizeof(char*));
    int len_arr = 0;
    int n = 0, m = 0;
    char* end = "Dragon flew away!";
    while (TRUE) {
        txt = readSentence();
        txt = delete_tab(txt);
        n++;

        if (extra(txt)) {
            m++;
            arr_txt[len_arr++] = txt;
        }

        if (len_arr == size_arr) {
            size_arr += DELTA;
            arr_txt = realloc(arr_txt, size_arr * sizeof(char*));
        }

        if (stop_input(end, txt)) {
            break;
        }
    }
    for (int i = 0; i < len_arr; i++) {
        puts(arr_txt[i]);
        free(arr_txt[i]);
    }
    free(arr_txt);
```

```
          printf("Количество предложений до %d и количество предложений
после %d", n - 1, m- 1);

          return 0;
     }
```