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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Программирование Си»
Тема: Использование указателей

Студент гр. 0382 _____ Крючков А.М.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Целью работы является освоение работы с указателями и динамической памятью.

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 – некоторая переменная, значением которой является адрес в памяти некоторого объекта, определяемого типом указателя. Для работы с указателями используется 2 оператора:

* – оператор разыменования;

& – оператор взятия адреса.

функция *malloc* (для ее использования следует

подключить заголовочный файл *stdlib.h*):

```
void * malloc( size_t sizemem ).
```

Функция выделяет из памяти *sizemem* и возвращает указатель на выделенную память, который следует привести к требуемому типу.

Для изменения размера выделенной ранее динамически области памяти используется функция realloc:

```
void * realloc( void * ptrmem, size_t size )
```

Она получает указатель на выделенную ранее область памяти и новый ее размер (он может быть как увеличен, так и уменьшен) и возвращает указатель на область памяти измененного размера (при изменении размера области памяти ее начальный адрес может измениться). Следовательно, функция *realloc* может выполняться в некоторых случаях довольно долго, поэтому не следует использовать ее слишком часто без явной необходимости.

Если выделенная память больше не требуется, следует обязательно высвобождать ее с помощью оператора *free*.

Формально в языке С нет специального типа данных для строк, но представление их довольно естественно. Строки в языке С – это массивы символов, завершающиеся нулевым символом *('\0')*. Это порождает следующие особенности, которые следует помнить:

– нулевой символ является обязательным;

– символы, расположенные в массиве после первого нулевого символа никак не интерпретируются и считаются мусором;

– отсутствие нулевого символа может привести к выходу за границу массива;

– фактический размер массива должен быть на единицу больше количества символов в строке (для хранения нулевого символа);

– выполняя операции над строками, нужно учитывать размер массива, выделенный под хранение строки;

– строки могут быть инициализированы при объявлении.

Выполнение работы.

Используется стандартная библиотека языка си, её заголовочные файлы *stdio.h, stdlib.h, string.h* для работы со строками и *ctype.h* для работы с символами.

Описание функций

int main() – объявлет массив *strs*, выделяет память под него. Затем выполняет функцию *get_text()*, выводит полученные предложения, их количество до и после, очищает память, с помощью фунции *free()*.

*char** get_text(char **strs)* – вводит предложения, используя функцию *get_string()*. Увеличивает размер массива предложений, используя *realloc*, если требуется. Считает количество всех предложений и предложений, у которых меньше 2 прописных букв. Завершается, когда вводится предложение *"Dragon flew away!"*.

char get_string(char *s)* - вводит предложения, используя функцию *getchar()*. Увеличивает размер массива символов, используя *realloc*, если требуется. Увеличивает *counterABC*, если введён символ верхнего регистра. Не добавляет в s символы *'\t', '\n'*. Завершает ввод при встрече *'.', ';', '?', '!'*.

Описание переменных

Глобальные переменные:

sizebefore – количество предложений в введенном тексте без учета последнего.

sizeafter – количество предложений в тексте без учета последнего после обработки.

counterABC – количество заглавных букв в текущем предложении.

c – переменная типа char, хранит последний введенный символ.

strsindex – количество предложений, которые были помещены в динамический массив предложений.

Переменные функции *main.c:*

*char** strs* - динамический массив предложений.

Переменные функции *get_text():*

*char** strs* - динамический массив предложений.

strsize – размер *strs*.

char s* – динамический массив, хранящий текущее предложение.

Переменные функции *get_string():*

char s* – динамический массив, хранящий текущее предложение.

Ssize - размер *s*.

sindex - количество символов, которые были помещены в динамический массив символов.

Выводы.

Была освоена работа с указателями и динамической памятью.

Разработана программа, выполняющая считывание и обработку текста.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: src/main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define StringSupply 50
#define TextSupply 20
char* get_string(char *s);
char** get_text(char **strs);
int sizebefore = 0, sizeafter = 0;
int counterABC = 0;
char c;
int strsindex = 0;

int main() {

    char **strs = malloc(20*sizeof(char*));
    strs = get_text(strs);
    for (int i = 0; i<strsindex; i++){
        printf("%s\n", strs[i]);
        free(strs[i]);
    }
    free(strs);
    printf("Количество предложений до %d и количество предложений посл
e %d", sizebefore-1, sizeafter-1);

    return 0;
}
char** get_text(char **strs){
    int strsize = TextSupply;
    while(1){
        if(strsindex==strsize){
            strsize+=TextSupply;
            strs = realloc(strs, strsize*sizeof(char*));

        }
        char* s = malloc(StringSupply*sizeof(char));
        s = get_string(s);
        if (s[0]==' ')
            s++;
        sizebefore++;
        if (counterABC<=1){
            strs[strsindex++]=s;
            sizeafter++;
        }
        if (!strcmp(s, "Dragon flew away!")) {
            break;
        }
    }
    return strs;
```

```
}
char* get_string(char *s){
    int ssize = StringSupply;
    int sindex = 0;
    counterABC = 0;
    while(1){
        if(sindex==ssize-1){
            ssize+=StringSupply;
            s = realloc(s, ssize*sizeof(char));

        }
        c = getchar();
        if (isupper(c))
            counterABC++;
        if (c!='\t' && c!='\n'){
            if (s!="" || c!=' '){
                s[sindex++]=c;
            }
        }
        if (c=='.' || c==';' || c=='?'){
            c=getchar();
            while (c=='\n')
                c = getchar();
            s[sindex]='\0';
            break;
        }
        if(c=='!'){
            s[sindex]='\0';
            break;
        }
    }
    return s;
}
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Nu555llam auctor vehicula dui, quis lobortis nibh. Vivamus sit amet viverra arcu, sed ultricies nulla. Dragon flew away!	2Nu555llam auctor vehicula dui, quis lobortis nibh. Vivamus sit amet viverra arcu, sed ultricies nulla. Количество предложений до 2 и количество предложений после 2	Программа работает правильно
2.	Dragon flew away!	Dragon flew away! Количество предложений до 0 и количество предложений после 0	Программа работает правильно
3.	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	11Nam 7ele mentum 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; Количество предложений до 12 и количество предложений после 11	Программа работает правильно

	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 lobortis nibh. Vivamus sit amet viverra arcu, sed ultricies nulla. Dragon flew away!		