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 Довченко М.К .

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Изучение процесса работы указателей и динамической памяти в С.

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).

* 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 – переменная, содержащая адрес какой либо другой переменной того же типа.

*- оператор разыменования.

&- оператор взятия адреса.

Функции для работы с динамической памятью:

- void* malloc (size)) – выделяет блок из size байт и возвращает указать на начало этого блока.
- void* calloc (num, size)) – выделяет блок для num элементов, каждый из которых занимает size байт и инициализирует все биты выделенного блока нулями.
- void* realloc(pointer, size)) – изменяет размер ранее выделенной области памяти на которую ссылается указатель и возвращает указатель на область памяти измененного размера.
- void* free(pointer) – высвобождает выделенную ранее память.

Выполнение работы.

1. Считывание данных.

Для считывания данных реализована функция char* readtext() с переменными

- char *text – строка символов на которую выделяется память с помощью функции malloc.
- size – равный константе SIZE = 50, используется для выделения и перераспределения памяти с помощью функций malloc и realloc.
- length – длина массива *text.
- c – переменная для считывания

в функции используется цикл while, считывающий поступающие данные посимвольно с помощью функции getchar() присваиваемой к переменной c, которая затем присваивается к символьному массиву text с индексом length. С каждой итерацией цикла length увеличивается на 1. Если length становится равным переменной size, происходит увеличение переменный size на константу ADD равной 50 и затем происходит перераспределение памяти ячеек памяти для массива text с увеличением количества ячеек. Цикл завершается когда переменная c становится равна одним из следующим символов “ ***. ; ? !*** “. После завершения цикла последнему элементу массива присваивается символ ***\0***. Функция возвращает символьный массив text содержащий одно предложение.

2. Обработка данных.

Для обработки данных используются две функции char* edit(char* e_str) и char* delete(char* del_str).

В функции edit используется цикл while который проверяет 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. После цикла символьный массив также проверяется с помощью функции delete. Функция возвращает символьный массив.

Функция delete проверяет соответствие предпоследнего элемента символьного массива и символа "**?**". Если условие выполняется, то символьный массив заменяется на "**\0**". Функция возвращает символьный массив.

3. Вывод данных и функция main

Функция main использует следующие переменные:

- char* terminalstr = "Dragon flew away!" данный символьный массив содержит последнюю строку вводимых данных.
- char** textarr – двумерный символьный массив, на который выделяется память с помощью функции malloc, содержащий весь вводимый текст.
- int arraylength – длина символьного массива textarr.
- char* str – символьный массив к которому присваивается значение функции readtext и edit.
- int counteraft. i– счетчики предложений
- int size_textarr – переменная к которой присваивается значение константы SIZE. Нужна для работы с функциями malloc и realloc.

В функции main содержится цикл while с помощью которого происходит считывание символов и сохранение их в массив str. Далее массив str обрабатывается с помощью ранее описанной функции edit. В цикле также содержится условие которое сравнивает строку из символьного массива textarr с индексом arraylength со строкой terminalstr. Если строки совпадают, цикл завершается. Если переменная arraylength становится равной переменной size_textarr, память выделенная на массив textarr ранее перераспределяется и увеличивается ее количество.

Следующий цикл for смещает элементы массива на -1 если они равны \0 (были удалены с помощью функции delete). Цикл завершается если текущий элемент массива с индексом i совпадает со строкой terminalstr.

Последний цикл выводит все строки двумерного массива textarr и освобождает выделенную на них память.

После этого на экран выводится количество предложений до обработки и после обработки, после чего выделенная память на массив textarr освобождается.

Исходный код файла см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	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; Donec accumsan convallis ipsum vitae lacinia. Dragon flew away!	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 8 и количество предложений после 6	Программа работает правильно.
2.	Nam 7elementum id enim eu congue; Aliquam at ultricies nisl, sed pretium nulla; Phasellus nunc augue, dApibus quis molestie iD, gravida a velit? Donec congue mauris sed lacus pulvinar, quis semper orci sol4licitudin? 40 Nu555lla rutrum feugiat felis a	Nam 7elementum id enim eu congue; Aliquam at ultricies nisl, sed pretium nulla; 40 Nu555lla rutrum feugiat felis a pharetra. Donec accumsan convallis ipsum vitae lacinia.	Программа работает правильно.

	pharetra. Donec accumsan convallis ipsum vitae lacinia. Vivamus sit amet viverra arcu, sed ultricies nulla. Ut auctor augue vel tincidunt tincidunt 555. Dragon flew away!	Vivamus sit amet viverra arcu, sed ultricies nulla. Ut auctor augue vel tincidunt tincidunt 555. Dragon flew away! Количество предложений до 8 и количество предложений после 6	
3.	Aenean sem ligula, laoreet ac sodales a, congue euismod neque; Aliquam 555 condimentum ligula arcu, non mollis ex pell555entesque finibus. Aenean sem ligula, laoreet ac sodales a, congue euismod neque; Maecenas 555 posuere velit efficitur, egestas nunc quis, dictum purus? Phasellus nunc augue, dApibus quis molestie iD, gravida a velit? Donec congue mauris sed lacus pulvinar, quis semper orci sol4licitudin? 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; Donec accumsan convallis ipsum vitae lacinia. Dragon flew away!	Aenean sem ligula, laoreet ac sodales a, congue euismod neque; Aliquam 555 condimentum ligula arcu, non mollis ex pell555entesque finibus. Aenean sem ligula, laoreet ac sodales a, congue euismod neque; 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 16 и количество предложений после 10	Программа работает правильно.

Вывод.

В ходе работы был изучен процесс выделения динамической памяти и использование указателей в языке С.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ФАЙЛОВ ПРОЕКТА

Файл lb3final.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define SIZE 50
#define ADD 50

char* readtext();
char* delete(char*);
char* edit(char*);

int main(){
    char* terminalstr = "Dragon flew away!";
    char** textarr = malloc(SIZE*sizeof(char*));
    int arraylength = 0, counteraft = 0, i;
    char* str;
    int size_textarr = SIZE;

    while(1){
        str = readtext();
        str = edit(str);
        if(!strcmp(textarr[arraylength++] = str, terminalstr)){
            break;
        }
        if(arraylength == size_textarr){
            size_textarr += ADD;
            textarr = realloc(textarr, size_textarr*sizeof(char*));
        }
    }
    for(i = 0; i <= arraylength;){
        if(!strcmp(textarr[i], terminalstr)){
            i++;
            break;}
        if(!strcmp(textarr[i], "\0")){
            for(counteraft = i; counteraft < arraylength; counteraft++){
```

```
                textarr[counteraft] = textarr[counteraft+1];}
        }
        else
            i++;

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

char* readtext(){
    int length = 0, size = SIZE, c;
    char *text = malloc(size*sizeof(char));
    while(1){
        c = getchar();
        text[length++] = c;
        if(length == size){
            size += ADD;
            text = realloc(text, size);
        }
        if(c == '.' || c == ';' || c == '?' || c == '!')
            break;
    }
    text[length]='\0';
    return text;
}

char* edit(char* e_str){
    int e_len = strlen(e_str), e_counter1;
    while(e_str[0] == ' ' || e_str[0] == '\t' || e_str[0] == '\n'){
        for(e_counter1 = 0; e_counter1 < e_len; e_counter1++){
            e_str[e_counter1] = e_str[e_counter1 + 1];
        }
    }
    e_str = delete(e_str);
    return e_str;
}

char* delete(char* del_str){
    if(del_str[(strlen(del_str)-1)] == '?'){
        del_str = "\0";
    }
    return del_str;
}

```