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 _____ Гудов Н.Р.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Освоение работы с указателями и динамической паматью.

Задание.

Вариант 1

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 есть цифра 7 (в любом месте, в том числе внутри слова), 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.

В работе были использованы такие конструкции языка Си как:

Функции printf/scanf для вывода и ввода значений на консоле. Getchar для считывания символа

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.

int strcmp (const char * str1, const char * str2) — сравнивает две строки;

Указатель - некоторая переменная, значением которой является адрес в памяти некоторого объекта, определяемого типом указателя.

Пользовательские функции для ввода и обработки входящих символов.

Выполнение работы.

Входные данные сохраняются и обрабатываются в двумерном массиве

Функция inpsent() считывает входные данные в массив предложений. Память выделяется динамически. Считывание посимвольное.

Переменные:

sentsize – определяет размер выделенной памяти;

sentlenght-определяет количество занятой памяти

sentence-ссылается на выделенный блок памяти для предложения

Функция space_c() для избавления от возможной табуляции в начале предложений. Ищет подряд идущие пробелы и табуляцию.

Функция no_seven() для отсеивания предложений, содержащих цифру 7. Изменяет значение переменной, и ,если оно не изменилось, внутри функции main предложение добавляется к остальным.

В каждой итерации цикла for при помощи функции printf() выводится очередное предложение. Блок памяти, занятый эти предложением освобождается в отдельном цикле. Далее при помощи функции free освобождается блок памяти, адрес которого записан в переменной text. В функции main вместе с обработкой полученных через консоль данных считается количество предложений в новом тексте и количество непрошедших проверку предложений из чего и выводится последняя строка.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	awdaw. Dragon flew away!	awdaw. Dragon flew away! Количество предложений до 1 и количество предложений после 1	Работает правильно
2.	Asdf7. Dragon flew away!	Dragon flew away! Количество предложений до 1 и количество предложений после 0	Работает правильно
3.	7. Dragon flew away!	Dragon flew away! Количество предложений до 1 и количество предложений после 0	Работает правильно
4	qwerty7. qwerty. Dragon flew away!	qwerty. Dragon flew away! Количество предложений до 2 и количество предложений после 1	Работает правильно

Выводы.

Была освоена работа с указателями и динамической памятью.

Разработана программа, выполняющая обработку считанного текста при помощи двумерного массива.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: menu3.c

```
#include <stdlib.h>
#include <string.h>
#define TRUE 1
#define FALSE 0

char* inpsent()
{
    int sentsize = 100;
    int sentlenght = 0;
    int sym;

    char *sentence = malloc(sentsize*sizeof(char));

    while (TRUE)
    {
        sym = getchar();
        sentence[sentlenght++] = sym;
        if (sym == '.' || sym == ';' || sym == '?' || sym ==
'!')

break;
        if (sentlenght == sentsize)
        {
            sentsize += 100;
            sentence = realloc(sentence, sentsize);
        }
    }
    if (sentlenght == sentsize)
    {
        sentsize += 1;
    }
    sentence[sentlenght] = '\0';
    return sentence;
}

char* space_c(char* sent)
{
    int i = 0;
    while(sent[i] == ' ' || sent[i] == '\t' || sent[i] == '\n')
    {
        int j;
        for (j = 0; j < strlen(sent) - 1; j ++)
        {
            sent[j] = sent[j+1];
        }

        sent[j] = '\0';
    }
```

```
    return sent;
}

int no_seven (char* sent)
{

    int ans = 1;
    for (int i = 0; i < strlen(sent)-1; i ++)
    {
        if (sent[i]=='7')
        {
            ans = FALSE;
        }
    }
    return ans;
}

int main()
{
    int text_len = 0;
    int wrongcount = 0;
    char* end_sent = "Dragon flew away!";

    int text_mem_size = 100;
    char** text = malloc(text_mem_size*sizeof(char*));

    char* sentence;

    while (TRUE)
    {
        sentence = inpsent();
        sentence = space_c(sentence);

        if (no_seven(sentence))
        {
            text[text_len++] = sentence;
        }
        else wrongcount += 1;

        if (text_len == text_mem_size)
        {
            text_mem_size += 100;
            text = realloc(text, text_mem_size*sizeof(char*));
        }

        if (!strcmp(sentence, end_sent)) break;
    }

    for (int i = 0; i < text_len; i++)
    {
        if (text[i][0]!='\0')
        {
            printf("%s\n", text[i]);
```

```
            }

        }
        for (int i = 0; i < text_len; i++)
        {
            free(text[i]);
        }
        free(text);

        printf("Количество предложений до %d и количество
предложений после %d\n", text_len+wrongcount-1, text_len-1);
        return 0;
    }
```