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. 9383 _____ Корсунов А. А.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2019

Цель работы.

Научиться работать с указателями, массивами и строками.

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).

Выполнение задания.

1) Объявления функции:

объявляются 2 функции, которые помогут в решении задачи.

2) Описание функции main():

- (int)transfer — количество созданных предложений;

- (int)senten — количество предложений;
- (char**)arr — указатель на двумерный массив;
- (int)size - количество предложений выделенных под текст;

в функции происходит вызов функции input(int* size, int* senten), которая возвращает указатель на двумерный массив, в котором лежит введенный текст, следом в transfer присваивается количество заполненных предложений, далее вызывается функция hatred_seven(char** arr, int* senten), которая изменяет массив согласно заданию. В конце происходит вывод и очистка выделенной памяти,

3) Описание функции input(int* size, int* senten):

- (int)index — индекс текущего символа в предложении;
- (int)count — индекс текущего предложения;
- (int)k — счетчик подряд идущих символов, формирующих «Dragon flew away!»;
- (int)check — условие выполнения считывания;
- (char*)end — указатель на массив из символов слова «Dragon flew away!»;
- (char**)arr — указатель на массив предложений текста;

В функции происходит создание массива предложений, пока не встретится терминальное предложение.

4) Описание функции hatred_seven(char arr, int* senten):**

- (int)index — индекс текущего символа в предложении;
- (int)count — индекс текущего предложения;

В функции происходит удаление предложений с символом „7“ и перевыделение памяти, если это необходимо;

Выводы.

В ходе проделанной работы была изучена работа с указателями, массивами и строками.

№ п/п	Входные данные	Выходные данные
1.	T. fdfhd; as 7tg fg? Dragon flew away!	T. fdfhd; Dragon flew away! Количество предложений до 3 и количество предложений после 2

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

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

char** input(int* size, int* senten);
void hatred_seven(char** arr, int* senten);

int main()
{
   int transfer;
   int senten = 5;
   char** arr;
   int size = 5;
   arr = input(&size, &senten);
   transfer = senten;
   hatred_seven(arr, &senten);
   for(int i = 0; i<senten; i++){
      printf("%s\n", arr[i]);
   }
     printf("Количество предложений до %d и количество предложений после %d", transfer-1, senten-1);
   for(int i = 0; i<size; i++){
      free(arr[i]);
   }
   free(arr);
   return 0;
}
```

```
char** input(int* size, int* senten){
    char letter_dragon;
    char letter;
    int index = 0;
    int count = 0;
    int k = 0;
    int check = 1;
    char* end = "Dragon flew away!";
    char** arr = (char**)calloc((*size), sizeof(char*));
    for(int i = 0; i < 5; i++){
        arr[i] = (char*)calloc(30, sizeof(char));
    }
    while(check == 1){
        letter = getchar();
        if(letter == '\n'){
            continue;
        }
        if((index == 0) && (letter == 'D')){
            if(((count+1) % 5 == 0) && (count != 0)){
                arr = (char**)realloc(arr, (count+6) * sizeof(char*));
                *size += 5;
                for(int i = count+1; i < count+6; i++){
                    arr[i] = (char*)calloc(30, sizeof(char));
                }
            }
            k = 0;
            arr[count][index] = letter;
            index++;
```

```
        k++;
        for(int i = 1; i < 17; i++){
            letter_dragon = getchar();
            arr[count][index] = letter_dragon;
            index++;
            if(letter_dragon == end[i]){
                k++;
            }
            else{
                k = 0;
                break;
            }
        }
        if(k == 17){
            check = 0;
            arr[count][index+1] == '\0';
        }
        continue;
    }
    if((index == 0) && (letter != 'D')){
        if(((count+1) % 5 == 0) && (count != 0)){
            arr = (char**)realloc(arr, (count+6) * sizeof(char*));
            *size += 5;
            for(int i = count+1; i < count+6; i++){
                arr[i] = (char*)calloc(30, sizeof(char));
            }
        }
        arr[count][index] = letter;
        index++;
```

```
            continue;
        }
        if((index != 0) && ((letter != ';') && (letter != '?') && (letter != '.'))){
            if(index % 30 == 0){
                arr[count] = (char*)realloc(arr[count], (index+30) * sizeof(char));
            }
            arr[count][index] = letter;
            index++;
            continue;
        }
        if((index != 0) && ((letter == ';') || (letter == '?') || (letter == '.'))){
            if(index % 30 == 0){
                arr[count] = (char*)realloc(arr[count], (index+2) * sizeof(char));
            }
            arr[count][index] = letter;
            arr[count][index+1] = '\0';
            getchar();
            index = 0;
            count++;
            continue;
        }
    }
    *senten = count+1;
    return arr;
}

void hatred_seven(char** arr, int* senten){
    int index = 0;
    int count = 0;
```

```
        while(1){
            if(arr[count][index] == '\0'){
                count++;
                index = 0;
                if(count == (*senten)){
                    break;
                }
                else{
                    continue;
                }
            }
            if(arr[count][index] != '7'){
                index++;
                continue;
            }
            if(arr[count][index] == '7'){
                for(int i = count; i < (*senten) - 1; i++){
                    if(strlen(arr[i]) < strlen(arr[i+1])){
                                arr[i] = (char*)realloc(arr[i], (strlen(arr[i+1])+1) *
sizeof(char));
                    }
                    strcpy(arr[i], arr[i+1]);
                }
                *senten = *senten - 1;
                index = 0;
            }
        }
    }
```