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 Осинкин Е.А.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучение работы с указателями и динамической памятью в языке Си.

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 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).

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.

В данной работе были использованы такие конструкции языка Си как:

- Функции заголовочного файла stdio.h:
  - int scanf (const char * format, ...) – функция считывает входные данные с консоли и помещает в переменную;
  - int printf (const char * format, ...) – функция выводит принимаемое значение на консоль;
- Функции заголовочного файла stdlib.h:
  - 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.

Кроме этого были использованы операторы if(){} else{}, for (){}, while(){}

Выполнение работы.

1. Функция *main*():

Объявляется указатель целочисленного типа *size* и указатель на указатель символьного типа *text*. После этого в целочисленную переменную *index* поступает результат работы функции *scan_text(&text, &size)*, а в целочисленную переменную count возвращаемое значение функции *delete_wrong_sentences(text, size, index)*. Далее в цикле *for* от переменной *i*, которая изменяется от 0 до *index*-1 с шагом 1, выводится строка из массива строк *text* под индексом *i*, но только при условии того, что первый символ этой строки не пустой символ. После этого с помощью функции *printf()* выводится информация о работе программы. Далее сначала освобождается

память, выделенная под каждую из строк из массива строк, а после этого и память, выделенная под хранение указателей на каждую строку.

2. Функция *scan_text*():

Функция *scan_text* принимает на вход указатель на указатель на указатель символьного типа *text* и указатель на указатель целочисленного типа *size*. В теле функции создаётся символьная переменная *chr*, целочисленная переменная *index*=0, указатель целочисленного типа *len*, в который помещается разыменованный указатель *size*. Далее с помощью функции *malloc()* выделяется блок памяти в *1*sizeof(int)* байта. Также создаётся указатель на указатель символьного типа *sent*, в которые помещается разыменованный указатель *text*. С помощью функции *malloc()* выделяется блок памяти размером *(i+1)*sizeof(*char)* байта. Далее с помощью оператора *while(c!='!')* выполняется ввод текста. С помощью функции *scanf()* пользователь вводит первый символ в предложении, и если этот символ не является символом табуляции, пропуском строки или пробелом, то начинается ввод остальной части предложения. В теле цикла *if* создаётся целочисленная переменная *number*, которая показывает, на сколько позиций в памяти относительно начала, сдвинуто положение. Переменная *sent_size=1* используется для указания длины строки. С помощью функции *malloc* выделяется блок памяти в *sent_size*sizeof(char*) байт для *index*-го предложения в массиве *sent. Sent[index][number]* присваивается значение *chr*. Далее, пока предложение не заканчивается, мы вводим *chr*, и если введенный символ не относится к символам табуляции, то *sent_size* и *number* увеличиваются на 1, и после этого блок памяти, выделяемый для *sent[index]* с помощью функции *realloc()*, расширяется на 1 символ. После того, как был получен знак конца предложения массив *sent[index]* расширяется ещё на один символ, чтобы вставить в конец строки \0 – знак, который передаёт компьютеру информацию о том, что строка закончена. В массив длин строк l в ячейку *len[index]* записывается длина строки *sent_size*, счётчик предложений *index* увеличивается на 1, массив предложений *sent* увеличивается до размера *(i+1)*sizeof(char*)*, массив l также увеличивается на 1. После того, как все предложения будут считаны в

разыменованный указатель на указатель целочисленного типа *size* помещается указатель *len*, а в разыменованный указатель на указатель на указатель символьного типа *text* помещается указатель на указатель символьного типа *sent*. Функция возвращает количество считанных предложений *index*.

3. Функция *delete_wrong_sentences()*

В функцию поступает указатель на указатель символьного типа *text*, указатель целочисленного типа *size* и целое число *index*. В целочисленную переменную *count*, в которой будет храниться количество предложений после обработки, помещается исходное количество предложений *index*. Далее с помощью двух циклов *for*, первый из которых проходит по массиву предложений *text*, а второй по каждой строчке *text[i]*, происходит проверка каждого символа и, если в предложении встречается цифра 7, то первый символ этого предложения становится \0, *count* уменьшается на 1, и с помощью оператора *break* завершаются циклы. Функция возвращает количество предложений после обработки *count*.
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.	qweqwe. qweqwe; eqw7qwe; qweqw? Dragon flew away!	qweqwe. qweqwe; qweqw? Dragon flew away! Количество предложений до 4 и количество предложений после 3	Программа работает правильно
2.	qwe7qwe. qwe3qwe; eqw7qwe; qweq7w? Dragon flew away!	qwe3qwe; Dragon flew away!	Программа работает правильно

		Количество предложений до 4 и количество предложений после 1	
3.	qwe7qwe. qwe3qwe; eqw7qwe; qweq7w? Dragon flew away!	qwe3qwe; Dragon flew away! Количество предложений до 4 и количество предложений после 1	Программа работает правильно

Выводы.

В ходе работы была изучена работа с динамической памятью и указателями.

Была разработана программа, считывающая с ввода текст и помещающая его в двумерный массив строк при помощи функции *scan_text*. Обработка данных происходит с помощью функции *delete_wrong_sentences*.

Обработанные данные возвращаются пользователю на консоль, занятая память под нужды программы освобождается.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла:

main.c

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

int scan_text(char*** text, int** size) {
     char chr = '\0';
     int index = 0;
     int* len = *size;
     len = (int*)malloc(1 * sizeof(int));
     char** sent = *text;
     sent = (char**)malloc((index + 1) * sizeof(char*));
     while (chr != '!') {
          scanf("%c", &chr);
          if ((chr != ' ') && (chr != '\t') && (chr != '\n')) {
               int number = 0;
               int sent_size = 1;
               sent[index] = (char*)malloc(sent_size *
sizeof(char));
               *(sent[index] + number) = chr;
               while ((chr != '.') && (chr != ';') && (chr !=
'?') && (chr != '!')) {
                    scanf("%c", &chr);
                    if ((chr != '\t') && (chr != '\n')) {
                         sent_size++;
                         number++;
                         sent[index] =
(char*)realloc(sent[index], sent_size * sizeof(char));
                         *(sent[index] + number) = chr;
                    }
               }
               sent[index] = (char*)realloc(sent[index],
(sent_size + 1) * sizeof(char));
               *(sent[index] + number + 1) = '\0';
               *(len + index) = sent_size;
               index = index + 1;
               sent = (char**)realloc(sent, (index + 1) *
sizeof(char*));
               len = (int*)realloc(len, (index + 1) *
sizeof(int));
          }
          *size = len;
          *text = sent;
     }
     return index;
}

int delete_wrong_sentences(char** text, int* size, int index) {
     int count = index;
```

```
    for (int i = 0; i < index; i++) {
        for (int j = 0; j<size[i]; j++) {
            if (text[i][j] == '7') {
                count--;
                text[i][0] = '\0';
                break;
            }
        }
    }
    return count;
}

int main() {
    int* size;
    char** text;
    int index = scan_text(&text, &size);
    int count = delete_wrong_sentences(text, size, index);
    for (int i = 0; i < index; i++) {
        if (text[i][0] != '\0') {
            printf("%s\n", text[i]);
        }
    }
    printf("Количество предложений до %d и количество предложений после
%d\n", index - 1, count - 1);
    for (int j = 0; j < index; j++) {
        free(text[j]);
    }
    free(text);
    return 0;
}
```