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. 9381 _____ Колованов Р.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Работа с указателями, динамической памятью и символьными массивами; получение навыков работы с ними.

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).

*** Порядок предложений не должен меняться**

*** Статически выделять память под текст нельзя**

*** Пробел между предложениями является разделителем, а не частью какого-то предложения**

Выполнение работы.

Для начала объявляются следующие переменные:

- ***n*** – хранит количество предложений в изначальном тексте.
- ***m*** – хранит количество предложений в отформатированном тексте.
- ***sentences*** – массив строк (указатель на массив, который хранит указатели на символьные массивы).
- ***maxSentencesSize*** – хранит размер массива ***sentences***.
- ***sentencesSize*** – хранит количество занятых ячеек массива ***sentences***.
- ***end*** – строка (символьный массив), содержащая предложение “Dragon flew away!”.

Помимо этого были написаны следующие функции:

- char* ***readSentence***(); - Функция считывает одно предложение, сохраняет его в динамической памяти и возвращает указатель на считанную строку. Ниже идет более подробное описание работы функции.
- int ***deleteSentencesContainingSeven***(char**, int); - Функция удаляет предложения, которые содержат символ ‘7’, и возвращает их количество. Ниже идет более подробное описание работы функции.
- void ***printText***(char**, int); - Функция выводит текст на экран.
- void ***cleanUpMemory***(char**, int); - Функция освобождает выделенную под хранения текста динамическую память.

Массив ***sentences*** создан в динамической памяти, поэтому во время выполнения программы есть возможность увеличивать его размер. Выделение памяти происходит при помощи функции *calloc()*.

Для начала выполняется посимвольное считывание текста в цикле *do-while* до тех пор, пока не встретится предложение *“Dragon flew away!”*. В цикле объявляются вызывается функция *readSentence()*.

Функция *readSentence()* содержит следующие переменные:

- ***symbol*** – хранит считанный символ.
- ***isInSentence*** – хранит значение 0 или 1 в зависимости от того, считываем ли мы само предложение или знаки между предложениями (требуется для удаления лишних знаков ‘\t’, ‘\n’ и ‘ ’ между предложениями).
- ***sentence*** – строка (массив символов).
- ***sentenceSize*** – хранит количество заполненных ячеек массива ***sentence***.
- ***maxSentenceSize*** – хранит размер массива ***sentence***.

Текущий цикл содержит вложенный цикл *do-while*, который посимвольно считывает одно предложение. Для начала проверяется, хватает ли места в массиве ***sentence*** для хранения очередного считанного символа. Если нет, то расширяем выделенную область памяти при помощи функции *realloc()*. Далее считанный символ заносится в массив ***sentence***. Считывание предложения происходит до тех пор, пока не встретится символы ‘?’, ‘!’, ‘.’ и ‘;’ (проверка происходит с помощью функции *strchr()*). В конец предложения добавляется нулевой символ. После того, как будет считано целое предложение, оно возвращается из функции и заносится в массив ***sentences***. Считывание предложений происходит до тех пор, пока последнее считанное предложение не станет равным строке ***end***. Для сравнивания строк используется функция *strcmp()*. После окончания ввода предложений переменным ***n*** и ***m*** присваивается значение переменной ***sentencesSize***.

Далее происходит вызов функции *deleteSentencesContainingSeven(),* которая пробегается по предложениям считанного текста и находит те из них, в которых есть символ *‘7’*. В случае, если в предложении есть символ *‘7’*, происходит удаление этого предложения: освобождается выделенная под это предложение динамическая память, указателю на это предложение присваивае значение *NULL*, тем самым показывая, что оно было удалено, а

после – значение m декрементируется. Проверка на вхождение символа *‘7’* происходит при помощи функции *strchr()*. Освобождение памяти происходит при помощи функции *free()*. Функция возвращает количество удаленных предложений, которое в дальнейшем вычитается из переменной ***m***.

В конце программа выводит отформатированный текст (каждое предложение на новой строке) при помощи функции *printText()*. После отформатированного текста выводится строка *“Количество предложений до n и количество предложений после m”*, где на место *n* и *m* подставляются значения переменных ***n*** и ***m***.

Далее происходит освобождение выделенной для программы памяти при помощи функции cleanUpMemory*()*.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1.	Hello. How are you? Dragon flew away!	Hello. How are you? Dragon flew away! Количество предложений до 2 и количество предложений после 2
2.	Hello. I'm 7. Here, do. Dragon flew away!	Hello. Here, do. Dragon flew away! Количество предложений до 3 и количество предложений после 2
3.	7? A 7. Seven. Dragon flew away!	Seven. Dragon flew away! Количество предложений до 3 и количество предложений после 1

Выводы.

Было изучена работа с указателями, динамической памятью, символьными массивами; получены навыки работы с ними.

Разработана программа, на вход которой подается текст, заканчивающийся предложением “Dragon flew away!”. Программа выводит отформатированный текст по следующему принципу:

- Каждое предложение начинается с новой строки.
- Табуляция в начале предложения удаляется.
- Все предложения, в которых есть цифра 7, удаляются.
- В конец текста добавляется предложение “Количество предложений до n и количество предложений после m”, где n - количество предложений в изначальном тексте и m - количество предложений в отформатированном тексте.

Для хранения текста использовались функции выделения, увеличения размера, освобождения динамической памяти *calloc(), realloc()* и *free(),* а также некоторые функции для работы со строками. В программе использовались операторы if-else, циклы for, while, do-while и указатели.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

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

char* readSentence() {
    char symbol, isInSentence = 0;
    int sentenceSize = 0, maxSentenceSize = 10;
            char*   sentence   =   (char*)calloc(maxSentenceSize,
sizeof(char));

    do {
        scanf("%c", &symbol);

         while (!isInSentence && strchr("\t\n ", symbol) != NULL)
{
            scanf("%c", &symbol);
        }

        isInSentence = 1;

        if (sentenceSize + 1 >= maxSentenceSize) {
            maxSentenceSize *= 2;
             sentence = (char*)realloc(sentence, maxSentenceSize *
sizeof(char));
        }

        sentence[sentenceSize++] = symbol;

    } while (strchr(";.?!", symbol) == NULL);

    sentence[sentenceSize++] = '\0';

    if (sentenceSize < maxSentenceSize) {
             sentence  =  (char*)realloc(sentence,  sentenceSize  *
sizeof(char));
    }

    return sentence;
}

int    deleteSentencesContainingSeven(char**    sentences,    int
sentencesSize) {
    int deletedSentencesCount = 0;

    for (int i = 0; i < sentencesSize; i++) {
        if (strchr(sentences[i], '7')) {
            deletedSentencesCount++;
            free(sentences[i]);
            sentences[i] = NULL;
        }
```

```
        }

        return deletedSentencesCount;
    }

    void printText(char** sentences, int sentencesSize) {
        for (int i = 0; i < sentencesSize; i++) {
            if (sentences[i] != NULL) {
                printf("%s\n", sentences[i]);
            }
        }
    }

    void cleanUpMemory(char** sentences, int sentencesSize) {
        for (int i = 0; i < sentencesSize; i++) {
            if (sentences[i] != NULL) {
                free(sentences[i]);
            }
        }

        free(sentences);
    }

    int main() {
        int n = 0, m = 0;
        int sentencesSize = 0, maxSentencesSize = 4;
              char**  sentences  =  (char**)calloc(maxSentencesSize,
sizeof(char*));
        char endSentence[] = "Dragon flew away!";

        do {
            if (sentencesSize >= maxSentencesSize) {
                maxSentencesSize *= 2;
                                sentences  =  (char**)realloc(sentences,
maxSentencesSize * sizeof(char*));
            }

            sentences[sentencesSize++] = readSentence();

        } while (strcmp(sentences[sentencesSize - 1], endSentence));

        if (sentencesSize < maxSentencesSize) {
              sentences = (char**)realloc(sentences, sentencesSize *
sizeof(char*));
        }

        n = sentencesSize;
        m = sentencesSize - deleteSentencesContainingSeven(sentences,
sentencesSize);

        printText(sentences, sentencesSize);
        printf("Количество предложений до %d и количество предложений
после %d\n", n - 1, m - 1);

        cleanUpMemory(sentences, sentencesSize);
```

```
    return 0;
}
```