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. 1304 _____ Лобанов Е.А.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2021

Цель работы.

Научиться пользоваться указателями и работе с текстом в языке Си.

Задание.

Вариант 2.

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.

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.

Выполнение работы.

Для выполнения задания была создана программа выполняющая считывание, обработку, и вывод текста.

С помощью *getchar()* и блочного выделения динамической памяти для каждого нового символа программа осуществляет запись предложений в *text[sentence]* до появления предложения "Dragon flew away!".

Если последний символ предложения оказывается равен ‘?’, тогда программа присваивает переменной *failed* значение 1 и освобождает память, выделенную для текущего предложения, для использования новым предложением.

Затем программа осуществляет вывод подходящих под условие предложений и вывод строки "Количество предложений до n и количество предложений после m", где n - количество предложений в изначальном тексте.

Во избежания утечки памяти программа освобождает всю динамически выделенную память.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	The quick. brown fox; jumps over? the lazy? dog.Dragon flew away!	The quick. brown fox; dog. Dragon flew away! Количество предложений до 5 и количество предложений после 3	Программа удалила все предложения с ‘?’ в конце.
2.	Lorem ipsum dolor sit amet; consectetur adipiscing elit. sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Dragon flew away!	Lorem ipsum dolor sit amet; consectetur adipiscing elit. sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Dragon flew away! Количество предложений до 3 и количество предложений после 3	Программа оставила текст без изменений ввиду отсутствия ‘?’ в конце.
3.	You just never quit, do you? Dragon flew away!	You just never quit, do you? Dragon flew away! Dragon flew away! Количество предложений до 1 и количество предложений после 0	Программа удалила все предложения с ‘?’ в конце.

Выводы.

Мы научились работать с указателями и освоили работу с текстом в языке Си.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define EOT "Dragon flew away!"

int main(){
      char **text;
      int total_count = 0, sentence = 0, count, state = 1, failed = 0;
      char c, temp;
      text = malloc(sizeof(char*));
      while (state){
            text = realloc(text, sizeof(char*) * (sentence + 2));
            text[sentence] = malloc(sizeof(char) * 2);
            count = 0;
            failed = 0;
            for (c = getchar(); c == '\t' || c == '\n' || c == ' '; c =
getchar()) {}
            text[sentence][count] = c;
            text[sentence][count+1] = '\0';
            while  ((c  !=  '.')  &&  (c  !=  ';')  &&  (c  !=  '?')  &&
(strcmp(text[sentence], EOT))){
                  count++;
                  c = getchar();
                  text[sentence] = realloc(text[sentence], sizeof(char) *
(count + 2));
                  text[sentence][count] = c;
                  text[sentence][count+1] = '\0';
            }
            if (c == '?')
                  failed = 1;
            if ((strcmp(text[sentence], EOT)) == 0)
                  state = 0;
            if (failed){
                  free(text[sentence]);
                  sentence--;
            }
            sentence++;
            total_count++;
      }
      for(int i = 0; i < sentence; i++)
            printf("%s\n", text[i]);

      printf("Количество  предложений  до  %d  количество  предложений
после %d\n", total_count-1, sentence-1);

     for(int i = 0; i < sentence; i++)
            free(text[i]);

      free(text);
      return 0;
}
```