МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Программирование»
Тема: “Использование указателей”

Студент гр. 7381 _____ Габов Е.С

Преподаватель _____ Берленко Т.А

Санкт-Петербург
2017

Целью данной работы является:
Написать 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, 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

Основные теоретические положения:
На вход программе подается строка символов. Она считывается с помощью функции getchar (считывает один символ). Если поступивший символ является “.” или “;” или “!”, то строка заканчивается. Эта строка помещается в массив указалетей num_str с помощью функции add_str (функция принимает на вход: указатель на указатель , порядковый номер строки , и строку которую надо поместить в num_str) . Далее по заданию удаляются пробелы и символы табуляции из начала строки.
Удаляются предложения в которых присутствует 7.(сначала предложения с 7 заменяются на предложения XXX, char** sort считывает только те предложения, которые не равны “XXX”).
Отсортированные предложения выводятся на экран с помощью команды puts. Текст заканчивается 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).

Память в данной лабораторной работе выделяется с помощью функций malloc(Функция malloc выделяет блок памяти, размером sizemem байт, и возвращает указатель на начало блока. Содержание выделенного блока памяти не инициализируется, оно остается с неопределенными значениями.) and realloc(Функция realloc выполняет перераспределение блоков памяти). В конце вся память очищается с помощью функции free(Функция free освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова malloc, calloc или realloc освобождается. То есть освобожденная память может дальше использоваться программами или ОС.)

Также в лабораторной работе используется функция strlen(функция возвращающая длину строки)

Вывод: В данной лабораторной работе изучается работа с указателями , в том числе динамическое выделение памяти , работа с этой памятью. Изучаются функции (malloc, realloc , calloc и функция free). Были написаны алгоритмы , позволяющие отформатировать текст следующим образом:
1)Вывести каждое новое предложение с новой строки
2)Удалить предложения с цифрой 7
3)Удалить табуляцию в начале предложения
4)Удалить пробелы в начале предложения

Функция main.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define size_str 10
char** add_str( char** , int , char* );
int main()
{
/*
  ввод строки с клавиатуры
  переделать её в текст
  отформатировать текст
*/
  char **num_str = 0 ; // pointer for first index
  char *str;
  int i=0;
  int k=0;
  int rem = 0;
  int dd=0;
  int con=2;
  char** sort=0;
  //enter text
  str = (char*) malloc( sizeof(char*) * 1 );
  sort = (char**)malloc( sizeof(char**) * 1000 );
  num_str = (char**)malloc( sizeof(char**) * 1000 );
  while( ( str[i] = getchar() ) != EOF)
    {
      str = realloc( str ,  sizeof(char*) * (i+2) );
      if ( str[i] == '.' || str[i] == ';' || str[i] == '?' || str[i] == '!' )
        {
          ++i;
          str[i] = '\0';
          num_str = add_str( num_str , k , str+rem );
          rem = i+1;
          k++;
        }
           i++;
    }
  // конец ввода текста
```

```
 //удаление табуляции
   for ( int j=0 ; j < k; j++ )
       if ( num_str[j][0] == '\t' )
         for ( int i=0; i < strlen(num_str[j]); i++ )
           num_str[j][i] = num_str[j][i+1];
 //конец удаления таб
 // удаление пробелов вначале
    for ( int j=0 ; j < k; j++ )
      while ( num_str[j][0] == ' ')
        for ( int i=0; i < strlen( num_str[j]) ; i++ )
           num_str[j][i] = num_str[j][i+1];
//  конец удаления пробелов
 // удаление предложений с 7 путем копирования
 for ( int j=0 ; j < k; j++ )
   for ( int q=0; q < strlen(num_str[j]) ; q++ )
     if ( num_str[j][q] == '7'  )
       num_str[j] = "xxx";
 for ( int b=0; b<k ; b++ )
    if  ( num_str[b] != "xxx")
      {
      sort[dd] = num_str[b];
        dd++;
      }
 for ( int pp=0; pp<dd; pp++ )
   puts( sort[pp] );
 // конец удаления 7
 printf("Количество предложений до %d и количество предложений после %d" , k-1 , dd-
1);
 free(sort);
 free(num_str);
 free(str);
 return 0;
}
char** add_str( char** str , int count , char *string )
 {
   if ( count == 0 )
     str = (char**)malloc( sizeof(char**) * ( count + 1 ) );
   else
     {
      char **copy = (char**)malloc(   sizeof(char*) * ( count + 1 ) );
      for ( int i = 0; i < count ; i++ )
        copy[i] = str[i];
      free(str);
      str = copy;
     }
   str[count] = (char*)malloc( sizeof(char*) * (strlen(string)+1) );
   strcpy( str[count] , string );
   return str;}
```

Makefile:

```
All: main.o
        Gcc main.o
```

```
Main.o: main.c
	Gcc –c main.c
```