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

ОТЧЕТ по лабораторной работе №1

по дисциплине «Программирование»

Тема: Указатели и динамические массивы.

Студент гр. 0382 _____ Ильин Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Изучить возможности указателей и способы работы с динамической памятью.

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, 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.

Память можно представить в виде некоторой ленты идущих друг за другом пронумерованных ячеек. Если с одномерным массивом кажется очевидным как он хранится в памяти, то работа с многомерными массивами может вызывать сложности.

Можно смотреть на двумерный массив (хотя это справедливо и для многомерных) как на массив массивов.

Это означает, что массив int arr[4][3] - это массив из 4 элементов, каждый из которых является тоже массивом из трех элементов.

Если считать, что массив располагается начиная с адреса 100, а тип int занимает 4 байта, то, например, адрес элемента arr[3][1] будет 140

Аналогично хранятся и массивы большей размерности. Так, массив arr[n][m][k] - это массив из n элементов, каждый из которых тоже массив из m элементов, элементы которого - массивы длины k

Для работы с динамической памятью используются следующие функции:

- malloc (void* malloc (size_t size)) - выделяет блок из size байт и возвращает указатель на начало этого блока
- calloc (void* calloc (size_t num, size_t size)) - выделяет блок для num элементов, каждый из которых занимает size байт и инициализирует все биты выделеного блока нулями
- 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.

Ход работы:

Для работы понадобились две библиотеки:

1) stdio.h — для ввода/вывода данных

2) stdlib.h — для динамической работы с памятью

В ходе работы были реализованы 5 функций, помимо main, это:

1) is_equal — получает на вход два предложения и сравнивает их, если предложения равны возвращает 0, иначе 1

2) text_lengh — получает на вход текст(массив ссылок на предложения (массивы char)) и последнюю строку этого текста, после чего считает количество предложений в тексте вплоть до конечного, после чего позвращает это количество.

3) newstr — считывает предложение по заданным параметрам, удаляет все символы пробеллов/табуляций перед предложениями и возвращает полученное предложение. В случае если при считывании встречается „?“ то возвращает предложение только из одной „?“. На вход получает предложение, после которого продолжать считывать ничего не надо.

4) mass — функция добавляет в переданный массив предложение полученное вызовом функции newstr, и если добавленное предложение конечное(на него должен заканчиваться текст), возвращает получившийся массив, иначе функция рекурсивно вызывает сама себя. На вход функция получает массив, в который нужно добавить предложения, конечное предложение, нынешнее считанное предложение, а также длинну нынешнего массива.

5) del_sent — на вход получает текст и его размер, из которого в цикле удаляет все предложения, состоящие только из „?“, после чего возвращает полученый текст.

В функции main создаётся изначально пустой массив, пустое предложение, задаётся конечная строка и поочерёдно вызываются нужные функции, после чего выводит результат всех этих функций на экран, добавляя строку с количеством предложений до и после изменения текста.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	Suspendisse quis mi neque7. Sed finibus magna et mauris elementum tempus? Sed finibus magna et mauris elementum tempus? Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra, per inceptos himenaeos. Donec at nunc ac mauris suscipit venenatis. Nam 7elementum id enim eu congue; Donec accumsan convallis ipsum vitae lacinia. Dragon flew away!	Suspendisse quis mi neque7. Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra, per inceptos himenaeos. Donec at nunc ac mauris suscipit venenatis. Nam 7elementum id enim eu congue; Donec accumsan convallis ipsum vitae lacinia. Dragon flew away! Количество предложений до 7 и количество предложений после 5	Программа работает верно
2.	. . . ... ;; ;?;.Dragon flew away!	. . . . . . ; ; ; ; . Dragon flew away! Количество предложений до 12 и количество	Программа работает верно

		предложений после 11	
3.	aezrggjfxh?sgrshsh?? Dragon flew away!	Dragon flew away! Количество предложений до 3 и количество предложений после 0	Программа работает верно

Выводы.

Была написана программа, которая получает на вход некоторый текст, после чего была разбита на предложения. Предложения в которых присутствовал „?“ были удалены, все символы перед предложениями были удалены, вместо них был вставлен символ переноса строки. Текст, который подаётся программе, после обработки, выводиться на экран, так же выводится дополнительное предложение, в котором написано сколько было изначально предложений и сколько осталось в конечном итоге.

Были изучены основные варианты работы с динамической памятью, а также принципы работы с указателями.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.c

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

int is_equal(char* sent, char* end_sent){
    int flag = 0;
    for (int i = 0;i<17;i++){
        if (sent[i + 1] != end_sent[i]){
            flag = 1;
            break;
        }
    }
    return flag;
}

int text_lengh(char ** mass, char * con){
    int i = 0;
    while(is_equal(mass[i], con)){
        i++;
    }
    i++;
    return i;
}

char* newstr(char* con){
    int step = 100;
    int sent_size = 0;
    char* sent;
    sent = malloc(100 * sizeof(char));
    char input_char = getchar();
    char b;
    if ((input_char != '\t') && (input_char != ' ') && (input_char != '\n')){
        if (sent_size + 10 > step){
            step += 100;
            sent = realloc(sent, step * sizeof(char));
        }
        sent[sent_size] = '\n';
        sent_size++;
    }
    else{
        while ((input_char == '\t') || (input_char == ' ') || (input_char == '\n')){
            input_char = '\n';
            b = getchar();
            if ((b == '\t')||(b == ' ')||(b == '\n')){
                input_char = b;
```

```
                                                                }
                                                               else{
                                                 if  (sent_size  +  10  >  step){
                                                                step  +=  100;
                                          sent  =  realloc(sent,  step  *  sizeof(char));
                                                                }
                                                   sent[sent_size]  =  input_char;
                                                               input_char  =  b;
                                                                  sent_size++;
                                                                }
                                                                }
                                                                }
    while ((input_char != '.') && (input_char != ';') && (input_char !
=                                                                       '?')){
               if  ((input_char  ==  '!')  &&  (~is_equal(sent,  con))){
                                                                break;
                                                                }
                                                               else{
                                                  if  (sent_size  +  10  >  step){
                                                                step  +=  100;
                                          sent  =  realloc(sent,  step  *  sizeof(char));
                                                                }
                                                   sent[sent_size]  =  input_char;
                                                       input_char  =  getchar();
                                                                  sent_size++;
                                                                }
                                                                }
                                             if       (input_char       ==       '?'){
                                            sent   =   malloc(2   *   sizeof(char));
                                                                sent[0]      =      '?';
                                                                }
                                                               else{
                                              if   (sent_size   +   10   >   step){
                                                               step   +=   100;
                                          sent  =  realloc(sent,  step  *  sizeof(char));
                                                                }
                                                    sent[sent_size]    =    input_char;
                                                                }
                                                          return            sent;

}

char ** mass(char ** input, char * con, char * nowstr, int kolstr){
                                             if       (is_equal(nowstr,       con)){
                                                              kolstr++;
                                  input  =  realloc(input,  kolstr  *  sizeof(char*));
                                                    nowstr     =     newstr(con);
                                               input[kolstr   -   1]   =   nowstr;
                                      return  mass(input,  con,  nowstr,  kolstr);
                                                                }
                                                               else{
                             input  =  realloc(input,  (kolstr  +  1)  *  sizeof(char*));
                                                    input[kolstr]     =     con;
                                                          return      input;
```

```
                                                                    }
}

char      **      del_sent(char**      text,      int      text_size){
                                    int         i         =         0;
                                while       (i       <       text_size){
                                      if     (text[i][0]     ==     '?'){
                                for  (int  j  =  i;  j  <  text_size;  ++j){
                                          text[j]   =   text[j   +   1];
                                                                    }
                                                          --text_size;
                                                                    }
                                                       else        {
                                                              i++;
                                                                    }
                                                                    }
                                                  return            text;
}

int                                  main()                                 {
             char    **    input,    **    start_text,    **    end_text;
                                  int        start_size,        end_size;
                        int      i      =      0,      j      =      0;
                                      char         *         input_sent;
                    input_sent     =     malloc(2     *     sizeof(char));
                      input     =     malloc(2     *     sizeof(char*));
                  char    *    end_sent    =    "Dragon    flew    away!";
              start_text   =   mass(input,   end_sent,   input_sent,   0);
                  start_size    =    text_lengh(start_text,    end_sent);
                   end_text     =     del_sent(start_text,     start_size);
                    end_size     =     text_lengh(end_text,     end_sent);
                        while      (is_equal(end_text[i],      end_sent)){
                                                    j       =       0;
           while  ((end_text[i][j]  !=  '.')  &&  (end_text[i][j]  !=  ';')){
                                       printf("%c",    end_text[i][j]);
                                                              j++;
                                                                    }
                                     printf("%c",     end_text[i][j]);
                                                              i++;
                                                                    }
    printf("\nDragon flew away!\nКоличество предложений до %d и коли-
чество   предложений   после   %d",   start_size-1,   end_size   -   1);
                                                  return            0;
}
```