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. 1304 _____ Кардаш Я. Е.

Преподаватель _____ Чайка К. В.

Санкт-Петербург
2021

Цель работы.

Изучение работы с динамической памятью и с указателями. Применение этих знаний на практике.

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 (\t, ' ') в начале предложения должна быть удалена.

Все предложения, в которых есть число 555, 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.

Сначала выполняется функция get_text, в которую передается указатель на массив строк text. Внутри этой функции выделяется определенный кусок памяти а затем выполняется функция get_sent, с помощь которой вводятся предложения (до тех пор, пока в get_sent не будет введено ключевое

предложение конца ввода. Если памяти не хватает она добавляется с помощью функции realloc

Внутри get_sent Так же сначала выделяется кусок памяти. Затем по заданию удаляются табы с помощью функции skip_tab. После вводится строка. Если памяти не хватает еона добавляется.

Затем происходит удаление предложений с ключевым словом (555) в функции del. В ней очищается память, выделенная на предложение с ключевым словом а массив смещается с помощь функции memmove. Проверка на ключевое слово происходит в функции check_del (наличие ключевого слова) и del_comb_in (проверка что это слово стоит отдельно).

После выполнения программы занятая память очищается с помощью функции free_text, а результат выводится на экран.

Экспериментальные результаты.

Входные данные	Выходные данные	Комментарий
Nulla facilisi. Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra, per inceptos himenaeos. 40 Nu555lla rutrum feugiat felis a pharetra. Sed finibus magna et mauris elementum tempus? Integer at quam et erat iaculis iaculis hendrerit a te4llus? Donec at nunc ac mauris suscipit venenatis. Sed finibus magna et mauris elementum tempus? Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi co7ndimentum 555 ex justo, nec pharetra mauris vestibulum a. Suspendisse quis mi neque7. 1 Vivamus eu nibh rhoncus, da456pibus ex non, sodales mi. Donec accumsan convallis ipsum vitae lacinia. Donec accumsan convallis ipsum vitae lacinia.	Nulla facilisi. Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra, per inceptos himenaeos. 40 Nu555lla rutrum feugiat felis a pharetra. Sed finibus magna et mauris elementum tempus? Integer at quam et erat iaculis iaculis hendrerit a te4llus? Donec at nunc ac mauris suscipit venenatis. Sed finibus magna et mauris elementum tempus? Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse quis mi neque7. 1 Vivamus eu nibh rhoncus, da456pibus ex non, sodales mi. Donec accumsan convallis ipsum vitae lacinia. Donec accumsan convallis ipsum vitae lacinia. Fusce finibus sapien magna, quis scelerisque ex sodales tristique. Nulla facilisi. Lorem ipsum dolor sit amet,	Тест успешный

Fusce finibus sapien magna, quis scelerisque ex sodales tristique. Nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Dragon flew away!	consectetur adipiscing elit. Dragon flew away! Количество предложений до 16 и количество предложений после 15	

Выводы.

Были изучены и применены на практике динамические массивы и указатели. Была написана программа считывающая динамически поступающий текст и обрабатывающая его.

Исходный код программы.

```
#INCLUDE <STDIO.H>
#INCLUDE <STRING.H>
#INCLUDE <STDLIB.H>
#DEFINE DEL_COMB "555"
#DEFINE BREAK_SENT "DRAGON FLEW AWAY!\N"
#DEFINE STAND 100
CHAR SKIP_TAB(){ //УДАЛЕНИЕ ТАБОВ В НАЧАЛЕ ПРЕДЛОЖЕНИЯ (ПРОПУСК)
      CHAR CHARACTER;
      FOR (CHARACTER = GETCHAR(); CHARACTER == ' ' || CHARACTER == '\T'
      || CHARACTER == '\N'; CHARACTER = GETCHAR()) {}
      RETURN CHARACTER;
}
VOID FREE_TEXT(CHAR*** TEXT,INT LEN){ //УДАЛЕНИЕ ВЫДЕЛЕННОЙ
ДИНАМИЧЕСКОЙ ПАМЯТИ
FOR (INT I=0;I<LEN;I++){
      FREE((*TEXT)[I]);
}
FREE(*TEXT);

}
CHAR* GET_SENT(){
      INT S_SIZE = STAND;
      CHAR *BUF = MALLOC(S_SIZE*SIZEOF(CHAR));
      IF (BUF) {
            CHAR *SENT = BUF;
            CHAR CHARACTER = SKIP_TAB();
            INT I;
```

```
                FOR (I = 0; CHARACTER != '.' && CHARACTER != ';' &&
CHARACTER != '?' && CHARACTER != '!'; CHARACTER = GETCHAR())
                {
                      SENT[I++] = CHARACTER;

                      IF (I == S_SIZE-2)
                      {
                            S_SIZE += STAND;
                            BUF = REALLOC(SENT, S_SIZE*SIZEOF(CHAR));
                            IF (BUF)
                            {
                                  SENT = BUF;
                            } ELSE
                            {
                                  FREE(SENT);
                                  RETURN NULL;
                            }
                      }
                }

                SENT[I++] = CHARACTER;
                SENT[I++] = '\N';
                SENT[I] = '\0';
                RETURN SENT;
          }
          RETURN NULL;

    }

    INT GET_TEXT(CHAR ***TEXT) {
          INT T_SIZE =STAND;
          CHAR** BUF = MALLOC(T_SIZE*SIZEOF(CHAR**));
          IF (BUF) {
                *TEXT = BUF;
                CHAR *SENT;
                INT I = -1;
                FOR (SENT = GET_SENT(); STRCMP(SENT, BREAK_SENT) != 0; SENT
                = GET_SENT())
                {

                      IF (SENT == NULL)
                      {
```

```
                        FREE_TEXT(TEXT, I);
                        RETURN 0;
                    }

                    (*TEXT)[++I] = SENT;
                    IF (I == T_SIZE) {
                        T_SIZE += STAND;
                        BUF = REALLOC(BUF, T_SIZE*SIZEOF(CHAR**));
                        IF (BUF)
                        {
                            (*TEXT) = BUF;
                        } ELSE
                        {
                            FREE_TEXT(TEXT, I);
                            RETURN 0;
                        }
                    }
                }

                (*TEXT)[++I] = SENT;
                I+=1;
                RETURN I;
            }
            RETURN 0;
        }

        INT DEL_COMB_IN(CHAR C){
            IF ((C < '0' || C > '9') && (C < 'A' || C > 'Z') && (C < 'A' || C
>           'Z')) {
            RETURN 1;}
            ELSE{
            RETURN 0;}

        }
        INT CHECK_DEL(CHAR* SENT){
            INT F;
            FOR(INT I=0; I<STRLEN(SENT)-STRLEN(DEL_COMB); I++){
                F=1;
                FOR (INT J=0;J<STRLEN(DEL_COMB);J++){
                    IF (SENT[I+J]!=DEL_COMB[J]){
                        F=0;
                        BREAK;
```

```
                        }

                }
                IF  (F&&((I==0||DEL_COMB_IN(SENT[I-
1]))&&(DEL_COMB_IN(SENT[I+STRLEN(DEL_COMB)])))){
                RETURN 1;}
          }
          RETURN 0;


    }

    VOID DEL(CHAR ***TEXT, INT* LEN){
          INT T_LEN=*LEN;
          INT I=0;
          WHILE(I<T_LEN){
                IF (CHECK_DEL((*TEXT)[I])){
                      FREE((*TEXT)[I]);
                      MEMMOVE(&(*TEXT)[I],&(*TEXT)[I+1],(T_LEN-
    I+1)*SIZEOF(CHAR*));
                      T_LEN--;
                }
                ELSE{
                      I++;
                }

          }
          *LEN = T_LEN;
    }


    INT MAIN(){
    CHAR** TEXT;
    INT LEN= GET_TEXT(&TEXT);
    INT LEN_AFT = LEN;
    DEL(&TEXT, &LEN_AFT);
    FOR (INT I=0;I<LEN_AFT;I++){
          PRINTF("%S",TEXT[I]);
    }
    PRINTF("КОЛИЧЕСТВО ПРЕДЛОЖЕНИЙ ДО %D И КОЛИЧЕСТВО ПРЕДЛОЖЕНИЙ ПОСЛЕ
%D", LEN-1,LEN_AFT-1);
```

```
        FREE_TEXT(&TEXT,LEN_AFT);
        RETURN 0;
}
```