**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

Студентка гр. 0382 _____ Чегодаева Е.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

**Цель работы.**

Изучение и использование указателей в языке Си.

**Задание.**

Вариант 3.

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 цифры внутри слов, должны быть удалены (это не касается слов, которые начинаются/заканчиваются цифрами). Если слово начинается с цифры, но имеет и цифру в середине, удалять его все равно требуется (4a4a).
- ✓ 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.**

Из библиотеки stdio.h: функции *getchar()* и *printf(),puts()* (Ввод и вывод соответственно).

Из библиотеки string.h: функция *strcmp*(<…>,<…>) (Функция побайтно сравнивает коды символов двух строк, на которые указывают аргументы функции. Сравнение продолжается до встречи первого отличающегося символа или пока не будут проверены все символы строк).

Из библиотеки ctype.h: функция isdigit() (Проверяет аргумент, является ли он десятичной цифрой), функция isalpha() (Проверяет аргумент, является ли он буквой).

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
- ✓ 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.

Операторы: *if(){} else{}, break;*

Циклы: *for (){}, while(){}.*

**Выполнение работы.**

1. Функция *main():*

Объявляются переменные *before* и *after*, которые будут хранить количество предложений изначального текста и текста после удаления необходимых приложений (изначально равны ноль). Далее с помощью указателей объявляются: *text* (хранит предложения из изначального текста) и *new_text* (хранит форматированный текст), выделяется память посредствам *malloc* (на один элемент). Затем посредствам цикла *while*, выходом из которого является встреча предложения “ Dragon flew away!”, происходит создание нового текста и подсчёт предложений → *text'*у присваивается предложение, полученное в функции *get_text()* и переменная *before* увеличивается на 1, далее с помощью оператора *if*

осуществляется проверка каждого приложения. Если флаг, полученный из функции *check_line*, имеет значение TRUE, то предложение добавляется к *new_text*, переменная *after* увеличивается на 1 и выделяется необходимая дополнительная память для данного предложения посредствам *realloc*. При помощи цикла *for ()* происходит вывод предложений из *new_text* посредствам *puts()* (последовательно выводится каждое предложение т.е. каждый элемент) и затем посредствам *printf()* выводится строчка "Количество предложений до n и количество предложений после m" - где n имеет значение *before-1*, а m - значение *after-1* (“-1” необходимо так как на последней итерации цикл *while* прибавит лишнюю единицу к данным числам).

2. Функция *get_text():*

Функция для ввода предложений.

Объявляется переменная *length* для хранения количества символов поступаемого предложения. Объявляется *sym* - для последующего ввода каждого символа. Объявляется *text* и выделяется память для него посредствам *malloc*, хранит всё предложение. Происходит считывание первого символа. Далее посредствам оператора *if* идёт проверка символа на пробел и табуляцию, если таковых нет - символ записывается в предложение и *length* увеличивается на 1. Далее реализован цикл *while*, выходом из которого является встреча одного из символов окончания предложения. Внутри цикла поступают символы и, вместе с тем, каждый символ посредствам *if* проверяется на табуляцию и перенос строки, если таковых нет - символ записывается в предложение, *length* увеличивается на 1 и выделяется необходимая память при помощи *realloc*. Затем в конце каждого предложения записывается символ конца строки. Функция возвращает предложение исходного текста без (возможно имеющихся) пробела в начале, табуляции и переноса строки.

3. Функция *check_line():*

Функция для проверки предложения на наличие слов, содержащих цифры в середине.

В функцию поступает предложение исходного текста. Переменной *flag* приваривается значение TRUE. Далее посредствам цикла *while*, где выходом является встреча символа окончания строки, проверяется каждое слово. Внутри цикла объявляются переменные *begin* (индекс начала слова, присваивается значение *i* цикла), *first_digit* (первая цифра), *first_alpha* (первая буква), *last_digit* (последняя цифра), *last_alpha* (последняя буква), *alpha_between* (буква в середине слова), *digit_between* (цифра в середине слова). Далее при помощи ещё одного цикла *while*, выходом из которого являются все возможные символы для разделения слов в предложении, а так же символы окончания предложения, находится номер последнего символа слова. Объявляется переменная *end* и в неё записывается индекс последнего элемента данного слова, определённое в предыдущем цикле и отнимается 1 (т.к. цикл на последней итерации добавит лишнюю 1). Затем проверяется первый и последний символ слова (цифра или буква) посредствам функций *isdigit(), isalpha()* и значение TRUE или FALSE присваиваться соответствующей переменой. Далее с помощью цикла *for()* проверятся каждый элемент (не включая первый и последний) на наличие цифры посредствам *isdigit()* и далее соответствующее значение присваивается переменным *alpha_between* и *digit_between.* После этого с помощью конструкции *if/else* реализуется проверка слова, если слово имеет цифру в середине → еще одним *if* происходит проверка длины слова, если слово состоит больше, чем из двух символов переменной *flag* присваивается значение FALSE и выход из цикла (так как это свидетельствуют о том, что слово с цифрой по середине уже есть в предложении). Далее, во избежание выхода за границу, при помощи *if* проверяются символы и при встрече символа окончания предложения происходит один шаг вперёд. Функция возвращает значение флага в *main().*

Разработанный программный код см. в приложении А.

**Тестирование.**

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

| № п/п | Входные данные | Выходные данные | Комментарии |
|---|---|---|---|
| 1. | Текст-пример с сайта e.moevm.info | Вывод совпадает. | Ответ верный. |
| 2. | Hi. Kak dela? hj8uj. Norm. Dragon flew away! | Hi.<br>Kak dela?<br>Norm.<br>Dragon flew away!<br>Количество предложений до 4 и количество предложений после 3 | Ответ верный. |
| 3. | Privet. fgh67gh? 7h7h7, KJKL. FGh67fgh. G6789g hythyy. Dragon flew away! | Privet.<br>Dragon flew away!<br>Количество предложений до 5 и количество предложений после 1 | Ответ верный. |
| 4. | Qwer5we. hohoh, l am Santa. J89j, goo? Dragon flew away! | hohoh, l am Santa.<br>Dragon flew away!<br>Количество предложений до 3 и количество предложений после 1 | Ответ верный. |
| 5. | A99a. Bbbbb. 6c6c6, lll. 7e, 8k. 78 ppp? qwe5we. k5k5k. Dragon flew away! | Bbbbb.<br>7e, 8k.<br>78 ppp?<br>Dragon flew away!<br>Количество предложений до 7 и количество предложений после 3. | Ответ верный. |

**Выводы.**

Были изучены указатели в языке Си.

Разработана программа с использованием указателей, динамической памяти и двумерного массива.

## ПРИЛОЖЕНИЕ А

## ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
char* get_text(){
     int length = 0;
    char sym = 0;
    char *text = malloc(1*sizeof(char));
    sym = getchar();
    if(sym!=' ' && sym!='\t'){
                text[length] = sym;
                length++;
     }
    while (sym != '.' && sym != ';' && sym != '?' && sym != '!'){
                sym = getchar();
                if (sym!='\t' && sym!='\n'){
                text[length] = sym;
                length++;
                text= realloc(text,(length+1)*sizeof(char*));
                }

   }
   text[length] = '\0';
   return text;
}

int check_line(char *text){
        int i=0;
        int flag = 1;
        while (text[i]!=';' && text[i]!='.' && text[i]!='?' && text[i]!='!'){
        int begin=i;
        int first_digit=0;
        int first_alpha=0;
        int last_digit=0;
        int last_alpha=0;
        int alpha_between=0;
        int digit_between=0;
        while(1){
                    if (text[i]==' ' | text[i]==','){
                        break;
                    }
                    if (text[i]==';' | text[i]=='.' | text[i]=='?'  | text[i]=='!'){
```

```
                    break;
                }
                    i++;
            }
            int end=i-1;
            first_digit=isdigit(text[begin]);
            first_alpha=isalpha(text[begin]);
            last_digit=isdigit(text[end]);
            last_alpha=isalpha(text[end]);
            for (int j=begin+1; j<end;j++){
                if (isdigit(text[j])){
                    digit_between=1;
                }
                else{
                    alpha_between=1;
                    }
                }
                if ((first_alpha && last_alpha && alpha_between && ! digit_between) || (first_digit && last_digit && digit_between && !alpha_between) || (first_digit && last_alpha && !digit_between && alpha_between) || (last_digit && first_alpha && !digit_between && alpha_between) || (first_digit && last_digit && ! digit_between && alpha_between)){
                }
                else{
                    if (end+1-begin>2){
                        flag = 0;
                        break;
                    }
                }
                if (text[i]!=';' && text[i]!='.' && text[i]!='?' && text[i]!='!'){
                    i++;
                }
            }
            return flag;
        }
        int main() {
           int before= 0;
           int after= 0;
           char* text;
           char** new_text = malloc(1*sizeof(char*));
           while (strcmp(text,"Dragon flew away!")){
                text = get_text();
                before++;
                if (check_line(text)==1){
                new_text[after] = text;
                after++;
```

```
            new_text = realloc(new_text, (after+1)*sizeof(char*));
            }
        }
        for (int i = 0; i <after ; i++){
            puts(new_text[i]);
        }
          printf("Количество предложений до %d и количество предложений после %d\n", before-1,after-1);
        return 0;
    }
```