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 _____ Павлов Д.Р.

Преподаватель _____ Чайка К.В.

Санкт-Петербург
2021

Цель работы.

Отработать навыки работы с указателями языке программирования Си. Научиться динамически выделять память и работать со строками.

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.

В данной лабораторной работе мы будем использовать такие полезные библиотеки как <stdio.h>, <string.h>, <stdlib.h>. Их мы и будем объявлять в самом начале. Они нам нужны для динманического выделения памяти и работы со строками. Так же я для удобства введу константу с названием END, в которой будет хранится конечное предложение нашего текста «Dragon flew away!\n». Важно отметить, что данное предложение заканчивается на символ

перевода строки, поскольку в дальнейшем я буду его добавлять в конец каждого предложения.

Далее мы объявляем нашу первую функцию «is_num_in_sentece». Аргументом которого будет указатель на предложение типа char. Если вкраце, то данная фугкция будет проверять есть ли число 555 в предложении. В теле данной функции мы задаем два условия, в которых будем проверять все существующие варианты того, как может быть представлено число 555 в данном предложении. Для этого в условиях мы будем использовать две функнции библиотеки string.h — strcmp и strstr(strcmp будет проверять варианты, при которых предложерие состоит только из числа 555; а strstr — есть ли число в принципе). Так же мы ввели еще одно условие, когда число 555 стоит в начале предложения, а далее идут пробелы окажется так же неверным для этого мы совместим strstr(sentence, „555 „) != NULL, и sentence[0-2] == „5“ оператором &&, а остальные - ||. Если все эти условия являются истинной, то функция возвращает число 1. Далее мы для удобства пишем еще одно условие, только на этот раз мы будем рассматривать случаи, в которых число стоит либо в конце, либо в середине предложения. Другими словами перед 555 обяхательно должен стоять пробел, а после — один из символов конца предложения, либо же просто пробел. В таком случае, если одно из условий подходит — мы возвращаем так же число 1. В противном случае работы данной функции мы возвращаем число 0.

Далее пишем функцию print_text, аргументом которой является двойной указатель на тип чар(текст, разделенный по предложениям). Эта функция по факту не делает ничего примечательного — мы просто пишем бексконечный цикл for(i = 0; i>=0; i++) внутри которой мы ставим условие if (is_num_in_sentence(text[i]) == 0) — мы выводим text[i](предложение). Другими словами, предложение будет выводится только при случае если в

нем нету числа 555. Далее мы пишем уже другое условие — if (strcmp(text[i]), END) == 0) — мы завершаем цикл. Эта строка значит, что если предложение (text[i]) и наше предложение конца текста (END) идентичны — происходит завершение.

Далее мы пишем еще одну ничем не примечательную функцию sentence_counter, аргументом которой все так же служит char** text. В теле мы объявляем переменную типа int равную нулу, которую назовем counter. Далее мы пишем все тот же бесконечный цикл, что и был в прошлой функции, который будет проходится по всем предложениям и считать их количество. Цикл заврешается если text[i] (предложение) равно конечному предложению. Функция возвращает значение counter.

Далее мы пишем похожую на предыдущую функцию sentence_counter_with_num, аргументом которого является все так же двойной укзаатель на тип чар (char** text). Эта функция будет считать количество предложений, в которых есть число 555. Первой строкой мы объявляем переменную типа int (counter) равную все так же нулю. Потом строим все так же бесконечный цикл for(i = 0; i>=0; i++) внутри которого записываем условие (if (is_num_in_sentecne(text[i]) == 1) => counter++) которое означает если в предложении есть число 555, то мы увеличиваем нашу переменную counter на 1. Далее вводим условие (if (strcmp(text[i], END) == 0) => break;) если наше предложение является конечным — завершаем цикл. В конце функции мы возвращаем значение counter.

Далее мы пишем функцию типа void которая будет освобождать память для нашего текста. В аргументе функции указываем двойной указатель типа char и I типа int(void free_text(char** text, int I)) в данном случае I — количество предложений в тексте. В теле функции пишем цикл for (int i = 0; i < I; i++)

которая будет выполнятся пока i меньше количества предложений. Внутри цикла мы пишем free(text[i]); что означает, что мы будем освобождать память из каждого одномерного массива(предложения) в тексте. После завращения цикла мы прописывем free(text), данная строка уже будет освобождать память из нашего двумерного массива(текста целиком) память. На этом работа данной функции закончена.

Далее мы пишем функцию форматирования нашего текста char** formating_text(char **text). Важно отметить, что данная функция будет возвращать двумерный массив, в котором будут хранится наши предложения. В теле функции мы сначала объявляем переменную типа int i = 0; эта переменная будет отвечать за индекс предложения в массиве. Далее мы прописывем цикл do-while(strcmp(text[i-1], END) != 0), что означает что цикл будет пробегаться до тех пор, пока наше предложение не станет равным конечному. Внутри цикла мы для начала объявляем переменную типа int, j, равную нулю — она будет отвечать за индекс символа в предложении. После мы в нашем двумерном массиве text делаем специальную функцию выделения памяти realloc (text = realloc(text, sizeof(char*) * (i+1)), которая в нашем двумерном массиве будет выделять память под каждые предложения. Далее мы присваиваем нашему предложению нулевое значение (text[i] = NULL) и делаем scanf(" "); для удаления табуляции. Далее мы все так же внутри нашего цикла заводим другой цикл, котороый уже на этот раз будет бесконечным (do — while(1)). Внутри него мы так же с помощью realloc выделяем посимовльно память для нашего предложения, при этом размер равен (j + 1) * sizeof(char), что означает что память будет выделяться для каждого символа в предложении (text[i] = realloc(text[i], sizeof(char) * (j + 1))). Далее мы заносим по индексу каждый элемент в массив — text[i][j] = getchar(). Далее мы делаем условие при котором наши символы перевода строки будут либо не заноситься в наше предложение, либо заменяться на

пробел. Замена на пробел будет осуществляться лишь в том случае, если предыдущий символ в предложении не равен пробелу. В противном случае мы просто делаем continue;. Далее мы прописывем условие которое будет выполнятся только в том случае, если данный символ равен символу конца предложения. (if ((text[i][j] == '.') || (text[i][j] == '?') || (text[i][j] == ';') || (text[i][j] == '!'))
). Если это условие выполняется, то тогда мы выделяем память для нашего предложения еще на +3 и дописывем в конец символ перевода строки и \0, после чего завершаем цикл.

Далее идет функция int main(). Внутри ее мы мы сначала объявляем три переменных — char** sentences = NULL (в ней будет хранится весь наш текст), int sents_count (количество предложений до), int sents_with_num (количество предложений после). Далее мы присваиваем переменной sentences возврат вызова функции при аргументе sentences, таким образом мы запишем наш текст, разделенный на предложения, в двумерный массив. Далее мы вызовем функцию print_text(sentences) — напишем текст. Далее мы присваиваем переменной sents_count, возвращаемое значение от функции sentence_counter при аргументе sentences. Далее мы приравниваем значение sents_with_num возвращаемое значение от функции sentence_counter_with_num с аргументом sentences. Далее мы выводи результат подсчета предложений — printf ("Количество предложений до %d и количество предложений после %d", sents_count-1, sents_count-1-sents_with_num);. А после — освобождаем память - free_text(sentences, sents_count
). В конце функция main возвращает 0.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	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. Fusce finibus sapien magna, quis scelerisque ex sodales tristique. Nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Dragon flew away!	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, consectetur adipiscing elit. Dragon flew away! Количество предложений до 16 и количество предложений после 15	True
2.	Po fwfwkfjwlfiw. dwjdwkhfwk 555 .	Po fwfwkfjwlfiw. kwjfkwfkw; Dragon flew away! Количество предложений	True

	kwjfkwfkw; fkwflwfwl 555? Dragon flew away!	до 4 и количество предложений после 2	
3.	jwvmwhvw. cwjkkjwcw? dlwdjw555? fwkjfnwkjfw Dragon flew away! Dragon flew away!	jwvmwhvw. cwjkkjwcw? dlwdjw555? fwkjfnwkjfw Dragon flew away! Dragon flew away! Количество предложений до 4 и количество предложений после 4	True
...			

Выводы.

В ходе выполнения лабораторной работы, было изучено как работать с указателями в языке Си, как работать с динамической памятью и строками.

Была разработана программа, которая по некторому принципу форматирует введеный текст.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define END "Dragon flew away!\n"


int is_num_in_sentence(char* sentence)
{
    if ((strcmp(sentence, "555.") == 0) || (strcmp(sentence, "555;") == 0) || (strcmp(sentence, "555?") == 0) || (((strstr(sentence, "555 ") != NULL)) && (sentence[0] == '5') && (sentence[1] == '5') && (sentence[2]) == '5')){
        return 1;
    }
     if ((strstr(sentence, " 555 ") != NULL) || ((strstr(sentence, " 555,")) != NULL) || (strstr(sentence, " 555.")) != NULL || (strstr(sentence, " 555?")) || ((strstr(sentence, " 555;")) != NULL)){
        return 1;
    }else{
        return 0;
    }
}



void print_text(char **text)
{
    int i;

    for (i = 0; i >= 0; i++){
        if (is_num_in_sentence(text[i]) == 0){
            printf("%s", text[i]);
        }

        if (strcmp(text[i], END) == 0){
            break;
        }
```

```
    }
}


int sentences_counter(char** text)
{
    int i;
    int counter = 0;

    for (i = 0; i>= 0; i++){
        if (strcmp(text[i], END) == 0){
            counter++;
            break;
        }else{
            counter++;
        }
    }

    return counter;
}

int sentence_counter_with_num(char **text)
{
    int i;
    int counter = 0;

    for (i=0; i>=0; i++) {
        if (is_num_in_sentence(text[i]) == 1){
            counter++;
        }

        if (strcmp(text[i], END) == 0) {
            break;
        }
    }

    return counter;
}


void free_text(char** text, int I){
```

```
        for (int i = 0; i < I; i++)
            free(text[i]);
        free(text);
    }


    char** formating_text(char **text)
    {
        int i = 0;

        do{
            text = realloc(text, sizeof(char *) * (i + 1));
            text[i] = NULL;
            scanf(" ");

            int j = 0;
            do {
                text[i] = realloc(text[i], (j + 1) * sizeof(char));
                text[i][j] = getchar();

                if ((text[i][j] == '\n') && (text[i][j-1] != ' ')) {
                    text[i][j] = ' ';
                }else if ((text[i][j] == '\n') && (text[i][j-1] == ' ')){
                    continue;
                }

                 if ((text[i][j] == '.') || (text[i][j] == '?') || (text[i]
[j] == ';') || (text[i][j] == '!')){
                    text[i] = realloc(text[i], (j + 3) * sizeof(char));
                    text[i][j+1] = '\n';
                    text[i][j+2] = '\0';
                    break;
                }
                j++;


            }while (1);

            i++;
        }while (strcmp(text[i-1], END) != 0);
```

```
    return  text;
}


int main()
{
    char **sentences = NULL;
    int sents_count;
    int sents_with_num;

    sentences = formating_text(sentences);
    print_text(sentences);
    sents_count = sentences_counter(sentences);
    sents_with_num = sentence_counter_with_num(sentences);


     printf("Количество предложений до %d и количество предложений после %d", sents_count-1, sents_count-1-sents_with_num);
    free_text(sentences, sents_count);

    return 0;
}
```