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. 0304 Мажуга Д.Р

Преподаватель Чайка К.В

Санкт-Петербург
2020

Цель работы.

Изучить принцип работы с указателями и динамической памятью. Изучить принцип обработки символьных массивов. Написать программу, считывающую текст и оперирующую с его предложениями.

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, в которых больше одной заглавной буквы, 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).

Основные теоретические положения.

Были использованы заголовочные файлы стандартной библиотеки: stdio.h, stdlib.h, string.h, ctype.h. Были использованы функции стандартной библиотеки void* malloc(size_t), void* realloc(void*, size_t) для расширения буфера для строк в случае нехватки места в нём; int getchar(), int isupper(int) для посимвольного ввода и удаления предложений с более чем одной заглавной буквой.

Выполнение работы.

1. Подключение заголовочных файлов stdio.h, stdlib.h, string.h, ctype.h. 2. Создание символических констант BUF_INIT_SIZE и BUF_INC_SIZE, обозначающих начальный размер буфера и шаг его увеличения соответственно.

3. Написание кода самой программы. Данный код состоит из двух вложенных циклов while. Первый, более широкий, цикл отвечает за пропуск пробельных символов (кроме символа новой строки) перед началом 2 предложения, и установку в начальное значение таких переменных, как: buf_pt – указатель на текущий элемент буфера, куда ведётся запись очередного символа; sent_loop – флаг вложенного цикла while, при установке которого в 0 данный цикл завершается; uppercase_cnt – количество букв верхнего регистра.

Вложенный же цикл while ответственнен за посимвольный ввод самого предложения, и обработку таких ситуаций, как: терминальное приложение (которое оканчивается восклицательным знаком), окончание ввода самого предложения (любым из трёх допустимых символов) и вывод его на экран; пропуск предложений, содержащих более одной заглавной буквы. При этом, для того, чтобы завершить выполнение всей программы после терминального предложения, используется флаг loop, который контролирует выполнение первого цикла while.

Для подсчёта количества предложений до и после работы программы используются переменные snt_cnt_bef и snt_cnt_aft соответственно.

Выводы.

Был изучен такой аспект языка программирования C, как работа с указателями, на примере работы со строками. В ходе написания программы был реализован буфер для чтения строчных данных, который увеличивается в размере по мере необходимости. Также в данной работе были задействованы и изучены функции заголовочного файла стандартной библиотеки ctype.h.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.c

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

#define BUF_INIT_SIZE 32
#define BUF_INC_SIZE 16

int main()
{
	size_t buf_sz = BUF_INIT_SIZE;
	char* buf = malloc(sizeof(char) * buf_sz);
	char* buf_pt;

	int c;
	int loop = 1, sent_loop;
	size_t uppercase_cnt;

	size_t snt_cnt_bef = 0, snt_cnt_aft = 0;
	while(loop)
	{
		buf_pt = buf;

		while( (c = getchar()) == ' ' || c == '\t') ;

		sent_loop = 1;
		uppercase_cnt = 0;
		while(sent_loop)
		{
			switch(c)
			{
				case '.': case ';': case '?': case '!':
					*buf_pt = c;
					++buf_pt;
```

```
                                        if(buf_pt >= buf + buf_sz){
                                                buf_sz += BUF_INC_SIZE;
                                                size_t buf_pt_diff = buf_pt - buf;
                                                buf = realloc(buf, sizeof(char) * buf_sz);
                                                buf_pt = buf + buf_pt_diff;
                                        }
                                        *buf_pt = '\0';
                                        sent_loop = 0;
                                        printf("%s\n", buf);
                                        if(c == '!'){
                                                loop = 0;
                                                --snt_cnt_bef;
                                                break;
                                        }
                                        ++snt_cnt_aft;
                                        break;
                                default:
                                        if(isupper(c)) ++uppercase_cnt;
                                        if(uppercase_cnt > 1){
                                                sent_loop = 0;
                                                while( (c = getchar()) != '.' && c != ';' && c != '?' && c !=
'!') ;
                                        }
                                        *buf_pt = c;
                                        ++buf_pt;
                                        if(buf_pt >= buf + buf_sz){
                                                buf_sz += BUF_INC_SIZE;
                                                size_t buf_pt_diff = buf_pt - buf;
                                                buf = realloc(buf, sizeof(char) * buf_sz);
                                                buf_pt = buf + buf_pt_diff;
                                        }
                                        c = getchar();
                                        break;
                                }
                        }
                        ++snt_cnt_bef;
                }
```

```
	printf("Количество предложений до %u и количество предложений после %u\n", snt_cnt_bef, snt_cnt_aft);
	free(buf);
	return 0;
}
```