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 _____ Стародубов М.В.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2021

Цель работы.

Отработать навыки работы с указателями в 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.

Сначала в файл подключаются стандартные библиотеки *stdlib.h*, *stdio.h*, *string.h*. Также определяются значения *BASE_SIZE* – "базовый" размер динамически выделяемой памяти, *LAST_SENTENCE* – последнее, «терминальное», предложение текста, *SPECIAL_COMBINATION* – комбинация символов, предложения содержащие которую должны быть удалены. Далее опишем работу всех функций.

1. Функция *get_first_character* – возвращает первый найденный символ строки, при этом пропуская пробелы, табуляции и символы переноса строки.

Используя фуднкцию *getchar*, получает символы из стандартного потока ввода, пока не встретит символ, отличный от символа табуляции, пробела bkb символа переноса строки. Возвращает данный символ.

2. Функция *get_sentence* – возвращает следующее предложение из *stdin*, в случае неудачи возвращает *NULL*.

В переменную *str_size* записывается значение *BASE_SIZE*, в «куче» динамически выделяется память для хранения *str_size* значений типа *char*, указатель на выделенную область памяти записывается в переменную *t*. Если удалось выделить необходимый объем памяти, то функция продолжает свою работу, иначе – возвращает *NULL*. В переменную *sentence* записывается указатель *t*, в переменную *character* записывается символ, полученный используя функцию *get_first_character* (таким образом удается удалить табуляцию в начале предложения). Далее рассмотрим цикл *for*. Цикл продолжается, пока символ, записанный в переменную *character*, не равен символу, означающему окончание предложения («.»/«;»/«?»/«!»), в конце каждой итерации в переменную *character* записывается новый символ, используя функцию *getchar*. В теле цикла символ, записанный в переменной *character* записывается в *i*-ый элемент списка *sentence* и значение счетчика

увеличивается на единицу, если после этого *i = str_size-2*, то значение *str_size* увеличивается на величину *BASE_SIZE*, и размер выделенной памяти, на которую указывает переменная *sentence*, расширяется так, чтобы в него могло поместиться *str_size* элементов типа *char*. После расширения выделенной памяти указатель на нее записывается в переменную *t*, если удалось увеличить размер выделенной памяти, то указатель *t* записывается в переменную *sentence*. В случае, если увеличить размер выделенной памяти не удалось, то ранее выделенная память освобождается, а функция возвращает *NULL*. После завершения работы цикла в конец массива *sentence* записываются последовательно символ, означающий конец предложения, символ перевода строки, символ конца строки. Функция возвращает указатель на массив *sentence*.

3. Функция *get_text* – записывает текст, получаемый из *stdin*, в *text*, возвращает количество считанных предложений, в случае неудачи возвращает 0.

Функция принимает на вход указатель на массив строк *text*. В переменную *text_size* записывается значение *BASE_SIZE*, в «куче» динамически выделяется память для хранения *text_size* значений типа указатель на указатель на *char*, указатель на выделенную область памяти записывается в переменную *t*. Если удалось выделить необходимый объем памяти, то функция продолжает свою работу, иначе – возвращает 0. В разименованную переменную *text* записывается указатель *t*, в переменную *sentence* будут записываться указатели на предложения. Перед началом выполнения цикла *for* в переменную *sentence* записывается предложение, получаемое с помощью функции *get_sentence*, цикл продолжается, пока в переменную *sentence* не будет записане терминальное предложение *«Dragon flew away!\n»*, в конце каждой итерации в переменную *sentence* с помощью функции *get_sentence* будет записано следующее предложение. В теле цикла сначало происходит проверка, удачно ли удалось

считать предложение, если предожение считать не удалось (*sentence = NULL*), то вся помять, выделенная под уже считанные предложения, а также под их хранение в массиве *text*, будет освобождена, а функция вернет 0. В противном случае выполнение функции продолжится, в *i*-й элемент массива *text* будет записано предложение из *sentence*, значение *i* увеличивается на единицу. Далее происходит проверка на необходимость выделения дополнительной памяти из «кучи». Если необходимо выделение памяти, то значение *text_size* увеличивается на величину *BASE_SIZE*, и размер выделенной памяти, на которую указывает переменная *text*, расширяется так, чтобы в него могло поместиться *text_size* элементов типа указатель на указатель на *char*. После расширения выделенной памяти указатель на нее записывается в переменную *t*, если удалось увеличить размер выделенной памяти, то указатель *t* записывается в *text*. В случае, если увеличить размер выделенной памяти не удалось, то вся память, выделенная под уже считанные предложения, а также под их хранение в переменной *text*, будет освобождена, а функция вернет 0. После завершения работы цикла в конец массива *text* записывается последнее считанное предложение, счетчик *i* увеличивается на единицу. Функция возвращает счетчик, значение которого равно количеству предложений в массиве *text*.

4. Функция *check_combination* – проверяет, является ли данный на вход функции символ частью слова или числа.

Функция принимает на вход символ *character*. Далее проверяется, является ли этот символ латинской буквой в любом регистре или цифрой, если не является, то функция возвращает 1, иначе – 0.

5. Функция *check_combination* – проверяет, содержит ли данное на вход предложение комбинацию символов *SPECIAL_COMBINATION*.

Функция принимает на вход предложение sentence. В переменную combination затисывается предложение, определенное в

SPECIAL_COMBINATION. Далее в двух вложенных циклах последовательно проверяется, входит ли подстрока *combination* в строку *sentence*. Если подсттрока была найдена, то с помощью функции *character_condition* происходит проверка символов в строке *sentence* перед и после встреченной комбинации. Если данные символы не являются частью слова или числа, то функция возвращает 0, иначе проверка продолжается. Если подстрока *combination*, не являющаяся частью слова или числа не была найдена в строке *sentence*, то функция возвращает 1.

6. Функция *delete_combination* – редактирует данный на вход массив *text*, удаляя все предложения, содержащие комбинацию символов *SPECIAL_COMBINATION*.

Функция принимает на вход указатель на массив строк *text* и указатель на количество предложений *text_size*. В «куче» динамически выделяется память для хранения отредактированного текста. Указатель на выделенную память записывается в переменную *t*. Если удалось выделить необходимый объем памяти, то выполнение функции продолжается. В переменную *out* записывается указатель на выделенную память, с помощью переменной *k* будет отслеживаться количество преложений, в которых не встречается комбинация *SPECIAL_COMBINATION*. Далее в цикле *for* каждое предложение из массива *text* проверяется на наличие комбинации *SPECIAL_COMBINATION* с помощью функции *check_combination*. Если комбинация не найдена, то предложение записывается в массив *out*, иначе память для хранения данного предложения освобождается. После выполнения цикла память для хранения указателей на пердложения *text* освобождается, в переменную *text* записывается новый указатель на отредактированный массив строк, в переменную *text_size* записывается количество предложений в отредактированном тексте.

7. Функция *free_text* – освобождение динамически выделенной памяти.

Функция принимает на вход указатель на массив предложений *text* и количество предложений в тексте *text_size*. Далее в цикле последовательно освобождается память для хранения предложений, сохраненных в массиве *text*, после выполнения цикла освобождается память для хранения массива *text*.

8. Функция *main*.

Создается переменная *text* лля хранения указателя на массив предложений. С помощью функции *get_text* в *text* записывается массив предложений из входных данных, размер данного массива записывается в переменную *text_size*. В переменную *new_text_size* копируется значение *text_size*. Массив *text* редактируется с помошью функции *delete_combination*, в переменную *new_text_size* записывается количество предложений в отредактированном тексте. Далее на экран последовательно выводятся все предложения из отредактированного текста. После этого выводится предложение с указанием количесва предложений в неотредактированном и отредактированним тексте. Далее с помощью функции *free_text* вся динамически выделенная память освобождается.

Выводы.

В ходе выполнения данной лабораторной работы было изучено, как работать с указателями в языке программирования Си. Было изучено, как работать с динамически выделяемой памятью, двумерными массивами и строками в языке программирования Си.

Была разработана программа, редактирующая данный на вход текст следующим образом:

- Каждое предложение начинается с новой строки.
- Табуляция (\t, ' ') в начале предложения удаляется.
- Все предложения, в которых есть число 555, удаляются.

•Текст заканчивается 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).

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

#define BASE_SIZE 50                                // "Базовый" размер динамически выделяемой памяти
#define LAST_SENTENCE "Dragon flew away!\n" // Последнее предложение текста
#define SPECIAL_COMBINATION "555"            // Комбинация символов, предложения содержащие которую должны быть удалены

char get_first_character();
char *get_sentence();
int get_text(char ***text);
int character_condition(char character);
int check_combination(char *sentence);
void delete_combination(char ***text, int *text_size);
void free_text(char ***text, int text_size);

int main()
{
          char **text;
          int text_size = get_text(&text);
          int new_text_size = text_size;

          delete_combination(&text, &new_text_size);

          for (int i = 0; i < new_text_size; i++)
          {
                    printf("%s", text[i]);
          }
          printf("Количество предложений до %d и количество предложений после %d\n", text_size-1, new_text_size-1);

          free_text(&text, new_text_size);

          return 0;
}

char get_first_character()
{   //   Возвращает   первый   найденный   символ   строки,   при   этом   пропуская пробелы/табуляции/символы переноса строки
          char character;
          for (character = getchar(); character == ' ' || character == '\t' || character == '\n'; character = getchar()) {}

          return character;
}

char *get_sentence()
{ // Получает следующее предложение из stdin, в случае неудачи возвращает NULL
```

```
	int str_size = BASE_SIZE;
	char *t = malloc(str_size*sizeof(char)); // Временная переменная для хранения адреса динамически выделяемой памяти

	if (t != NULL)
	{
		char *sentence = t;
		char character = get_first_character();
		int i;

		for (i = 0; character != '.' && character != ';' && character != '?' && character != '!'; character = getchar())
		{
			sentence[i++] = character;

			if (i == str_size-2)
			{ // Условие необходимости выделения дополнительной памяти
				//	[a_0,	a_1,	a_2,	...,	a_(str_size-2), a_(str_size-1), a_(str_size)]
				//                                                  ^'\n' ^'\0'
				str_size += BASE_SIZE;
				t		=		realloc(sentence, str_size*sizeof(char));
				if (t != NULL)
				{
					sentence = t;
				} else
				{
					free(sentence);
					return NULL;
				}
			}
		}

		sentence[i++] = character; // Добавление символа, означающего окончание стороки ('.'/';'/'?'/'!')
		sentence[i++] = '\n';		// Добавление символа переноса строки
		sentence[i] = '\0';		// Добавление символа конца строки

		return sentence;
	}
	return NULL;
}
int get_text(char ***text)
{ // Записывает текст, получаемый из stdin, в text, возвращает количество считанных предложений;
  // В случае неудачи возвращает 0
	int text_size = BASE_SIZE;
	char **t = malloc(text_size*sizeof(char**)); // Временная переменная для хранения адреса динамически выделяемой памяти

	if (t != NULL)
	{
```

```
                *text = t;
                char *sentence;
                int i = 0;

                for    (sentence    =    get_sentence();    strcmp(sentence,
LAST_SENTENCE) != 0; sentence = get_sentence())
                {
                        if (sentence == NULL)
                        {
                                free_text(text, i);
                                return 0;
                        }

                        (*text)[i++] = sentence;

                        if (i == text_size)
                        { // Условие необходимости выделения дополнительной
памяти
                                text_size += BASE_SIZE;
                                t = realloc(t, text_size*sizeof(char**));
                                if (t != NULL)
                                {
                                        (*text) = t;
                                } else
                                {
                                        free_text(text, i);
                                        return 0;
                                }
                        }
                }

                (*text)[i++] = sentence;

                return i;
        }
        return 0;
}

int character_condition(char character)
{ // Проверяет, является ли символ частью слова/числа
        if ((character < '0' || character > '9') && (character < 'A' ||
character > 'Z') && (character < 'a' || character > 'z'))
        {
                return 1;
        }
        return 0;
}

int check_combination(char *sentence)
{ // Проверяет, содержит ли предложение комбинацию символов SPECIAL_COMBINATION
        char *combination = SPECIAL_COMBINATION;
        int flag;

        for (int i = 0; i < strlen(sentence) - strlen(combination); i++)
        {
                flag = 1;
                for (int j = 0; j < strlen(combination); j++)
```

```
                {
                        if (sentence[i+j] != combination[j])
                        {
                                flag = 0;
                                break;
                        }
                }

                if (flag && ((i == 0 || character_condition(sentence[i-1]))
&& character_condition(sentence[i+strlen(combination)])))
                { // Проверка, являются ли символы перед и после найденой
комбинации частью слова/числа
                        return 0;
                }
        }
        return 1;
}
void delete_combination(char ***text, int *text_size)
{ // Редактирует массив text, удаляя все предложения, содержащие комбинацию
символов SPECIAL_COMBINATION

        char **t = malloc(*text_size*sizeof(char**)); // Временная переменная
для хранения адреса динамически выделяемой памяти

        if (t != NULL)
        {
                char **out = t; // Массив, в котором будут записаны
предложения, не содержащие комбинацию SPECIAL_COMBINATION
                int k = 0;

                for (int i = 0; i < *text_size; i++)
                {
                        if (check_combination((*text)[i]))
                        {
                                out[k++] = (*text)[i]; // В случае, если
предложение содержит комбинацию SPECIAL_COMBINATION, то предложение добавляется
в out
                        } else
                        {
                                free((*text)[i]);       // Иначе, память,
выделенная под предложение, освобождается
                        }
                }

                // Переписывание содержимого массива text
                free(*text);
                *text = out;
                *text_size = k;
        }
}
void free_text(char ***text, int text_size)
{ // Освобождение памяти
        for (int i = 0; i < text_size; i++) {
                free((*text)[i]);
        }
        free(*text);
}
```

ПРИЛОЖЕНИЕ В

ТЕСТИРОВАНИЕ

Данные тестирования представлены в таблице 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	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	Результат корректен

	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!	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	
2.	Nulla facilisi. Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra, per inceptos himenaeos. 40 Nu 555 lla 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	Nulla facilisi. Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra, per inceptos himenaeos. 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?	Результат корректен

	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 lacin555 ia. Donec accumsan convallis ipsum vitae lacinia. Fusce finibus 555sapien magna, quis scelerisque ex sodales tristique. Nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Dragon flew away!	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 lacin555 ia. Donec accumsan convallis ipsum vitae lacinia. Fusce finibus 555sapien 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 14	