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. 0382 _____ Злобин А. С.

Преподаватель _____ Чайка К. В.

Санкт-Петербург
2020

Цель работы.

Изучение работы с указателями и динамической памятью в языке Си.

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 есть цифра 7 (в любом месте, в том числе внутри слова), 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**

Основные теоретические положения.

В данной лабораторной работе были использованы следующие конструкции языка С:

- Функции библиотеки stdio.h:
  - printf()–функция выводит на консоль значине аргумета
  - getchar()–считывает символ из стандартного потока ввода
- Функция библиотеки stdlib.h:
  - 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.
  - free (void free (void* ptr)) - высвобождает выделенную ранее память
- Циклы:
  - while(){}–каждая итерация проверяет, выполняется ли условие в круглых скобках, если оно верно, то выполняется код в фигурных скобках, а если неверно, то происходит выход из цикла
  - for(){<переменная>; <условие>; <выражение_1>}–код в теле цикла будет исполнятся до тех пор, пока объявленная в цикле переменная будет удовлетворять условию цикла, выражение_1 каким-либо способом меняет значение этой переменной
- Операторы:
  - if(){} ... else{}–если выполняется условия, указанное в круглых скобках, то выполняется код в фигурных скобках после if, иначе–в фигурных скобках после else(elsene является обязательной частью конструкции)
- Функции:
  - <тип_функции> имя_функции(<аргумент_1>, ... , <argument_n>) {}–при вызове данной функции в главной(main) функциивыполняется

код в фигурных скобках, а затем возвращает значение оператором return(если тип функции не void)

Выполнение работы.

В начале программы необходимо подключить следующие библиотеки:

- stdio.h–используется для подключения ввода-вывода (printf(), scanf())
- stdlib.h–используются для доступа к функции abs(), которая позволяет получить модуль числа

1. Функция main():

В начале объявляем переменную text, которая является указателем на указатель типа char. Она будет отвечать за хранение текста. Переменная size хранит размер считанного текста, а переменная edited_size — размер отредактированного текста. Далее вызывается функция get_text(), после чего по указателю text будет доступен массив предложений текста. Затем вызывается функция edit_text(), которая скроет все предложения содержащие цифру 7. Вывод осуществляется по предложениям с помощью функции printf() в цикле for. Затем очищается память каждого предложения текста и массива, который содержал ссылки на предложения.

2. Функция get_text()

Функция принимает указатель на массив предложений, куда должен быть сохранён текст. В функции создаётся переменные типа int: size_sentence, size_text которые хранят размер текущего предложения и считываемого текста соответственно. Далее с помощью функции malloc выделяется память под первое предложение в тексте. В цикле do while происходит поочерёдное считывание предложения и выделение памяти под следующее с помощью функции realloc(). В конце цикла происходит копирование ссылки на текст в переменную text, которая доступна из функции main(). Цикл выполняется до тех пор, пока последний символ в последнем предложении не равен „!“, т. к. по условию только последнее предложение может заканчиваться этим символом. Функция возвращает количество предложений в ведённом тексте.

3. Функция get_sentence()

Функция принимает на вход указатель на массив символов, который является предложением, и сохраняет введённое предложение по этому указателю. В начале объявляются переменная sent_size типа int, которая хранит размер предложения и переменная с типа char, которая хранит текущий символ. Далее объявляется указатель на char, который будет являться началом предложения. В цикле do while происходит посимвольное считывание с консоли до тех пор, пока не появится первый символ отличный от пробела, \t, и \n. Далее выделяется память под первый символ и символ с сохраняется в эту выделенную память. Далее происходит посимвольное считываение, увеличение выделенного объёма памяти с помощью realloc и сохраниние символов в предложение. Когда будет достигнут конец предложения, в последнюю ячейку созранится смвол конца строки и ссылка на новое предложение будет сохранена в переданную ссылку. Функция возвращает размер введённого предложения.

3. Функция edit_text()

В функции с помощью вложенных циклов for записывается символ конца строки в начало предложения в тех предложениях , где есть цифра 7. Функция возвращает размер изменённого текста.

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.	Aenean magna massa, scelerisque quis sagittis at, pharetra a lectus? Aliquam at ultricies nisl, sed pretium nulla; Lorem ipsum dolor sit amet,	Aenean magna massa, scelerisque quis sagittis at,pharetra a lectus? Aliquam at ultricies nisl,sed pretium nulla; Lorem ipsum dolor sit	Программа работает верно

	consectetur adipiscing elit; Nam 7elementum id enim eu congue; Ut a7uctor, leo eu dictum vestibulum, tortor enim consequat mauris, eget consectetur justo quam et 7 metus. Nulla facilisi.Dragon flew away!	amet,consectetur adipiscing elit; Nulla facilisi. Dragon flew away! Количество предложений до 6 и количество предложений после 4	

Выводы.

В ходе работы была изучена работа с динамической памятью и указателями. Была разработана программа, считывающая с ввода текст и помещающая его в двумерный массив строк при помощи функциий get_text и get_sentence. Обработка данных происходит с помощью функции edit_text. Обработанные данные возвращаются пользователю на консоль, занятая память под нужды программы освобождается.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

int get_sentence(char ** received_sentence)
{
	int sent_size=1;
	char c;
	char *sentence = *received_sentence;
	do
	{
		c = getchar();
		if (c!=' ' && c!='\t' && c!='\n')
		{
			sent_size=1;
			sentence=malloc(sent_size*sizeof(char));
			*(sentence + sent_size - 1)=c;
			while (c!='.' && c!=';' && c!='?' && c!='!')
			{
				c = getchar();
				if (c!='\t' && c!='\n')
				{
					sent_size+=1;
					sentence	=	realloc(sentence,
sent_size*sizeof(char));
					*(sentence+sent_size - 1)=c;
				}

			}
			sentence		=		realloc(sentence,
(sent_size+1)*sizeof(char));
			*(sentence + sent_size)='\0';
		}
	}while (c!='.' && c!=';' && c!='?' && c!='!');
	*received_sentence = sentence;
	return sent_size-1;
}

int get_text(char*** text)
{
	int size_sentence = 0;
	int size_text=0;
	char **sent=*text;
	sent=malloc((size_text+1)*sizeof(char*));
	do
	{

			size_sentence = get_sentence(&sent[size_text]);
			size_text = size_text + 1;
			sent=realloc(sent,(size_text + 1)*sizeof(char*));
```

```
            *text=sent;
        }while (*(sent[size_text-1]+size_sentence)!='!');
        return size_text;
    }

    int edit_text(char** text, int size)
    {
        int i=0;
        int j=0;
        int changed_size=size;
        for (i=0; i<size; i++)
        {
            for (j=0; text[i][j]!='\0'; j++)
            {
                if (text[i][j]=='7')
                {
                    changed_size--;
                    text[i][0]='\0';
                    break;
                }
            }
        }
        return changed_size;
    }

    int main()
    {
        char **text;
        int size;
        int edited_size;
        int i;
        size = get_text(&text);
        edited_size = edit_text(text, size);
        for (i=0; i<size; i++)
        {
            if (text[i][0]!='\0')
            {
                printf("%s\n", text[i]);
            }
        }
        printf("%s %d %s %d\n", "Количество предложений до", size-1,
"и количество предложений после", edited_size-1);
        for (int j=0;j<size;j++)
        {
            free(text[j]);
        }
        free(text);
        return 0;
    }
```