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 _____ Сергеев Д.А.

Преподаватель _____ Жангиров Т.Р.

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 работе были использованы такие конструкции языка Си как:

- Функции заголовочного файла stdio.h:
  - int scanf (const char * format, ...) – функция считывает входные данные с консоли и помещает в переменную;
  - int printf (const char * format, ...) – функция выводит принимаемое значение на консоль;
- Функции заголовочного файла 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.

Кроме этого были использованы операторы if(){} else{}, for (){}, while(){}

Выполнение работы.

1. Функция *main*():

Объявляется указатель целочисленного типа (*len*) и указатель на указатель символьного типа (*s*). После этого в целочисленную переменную *i* поступает результат работы функции *text(&s, &len)*, а в целочисленную переменную count возвращаемое значение функции *delete_wrong_sentences(s,len,i)*. Далее в цикле *for* от переменной *t*, которая изменяется от 0 до *i*-1 с шагом 1, выводится строка из массива строк *s* под индексом *t*, но только при условии того, что первый символ этой строки не пустой символ. После этого с помощью функции *printf()* выводится информация о работе программы. Далее сначала освобождается

память, выделенная под каждую из строк из массива строк, а после этого и память, выделенная под хранение указателей на каждую строку.

2. Функция *text*():

Функция *text* принимает на вход указатель на указатель на указатель символьного типа *s* и указатель на указатель целочисленного типа *len*. В теле функции создаётся символьная переменная *c*, целочисленная переменная *i*=0, указатель целочисленного типа *l*, в который помещается разыменованный указатель *len*. Далее с помощью функции *malloc()* выделяется блок памяти в *1*sizeof(int)* байта. Также создаётся указатель на указатель символьного типа *sent*, в которые помещается разыменованный указатель *s*. С помощью функции *malloc()* выделяется блок памяти размером *(i+1)*sizeof(*char)* байта. Далее с помощью оператора *while(c!='!')* выполняется ввод текста. С помощью функции *scanf()* пользователь вводит первый символ в предложении, и если этот символ не является символом табуляции, пропуском строки или пробелом, то начинается ввод остальной части предложения. В теле цикла *if* создаётся целочисленная переменная *number*, которая показывает, на сколько позиций в памяти относительно начала, сдвинуто положение. Переменная *sent_size=1* используется для указания длины строки. С помощью функции *malloc* выделяется блок памяти в *sent_size*sizeof(char*) байт для *i*-го предложения в массиве *sent. Sent[i][number]* присваивается значение *c*. Далее, пока предложение не заканчивается, мы вводим *c*, и если введенный символ не относится к символам табуляции, то *sent_size* и *number* увеличиваются на 1, и после этого блок памяти, выделяемый для *sent[i]* с помощью функции *realloc()*, расширяется на 1 символ. После того, как был получен знак конца предложения массив *sent[i]* расширяется ещё на один символ, чтобы в вставить в конец строки \0 – знак, который передаёт компьютеру информацию о том, что строка закончена. В массив длин строк l в ячейку *l[i]* записывается длина строки *sent_size*, счётчик предложений i увеличивается на 1, массив предложений sent увеличивается до размера *(i+1)*sizeof(char*)*, массив l также увеличивается на 1. После того, как все предложения будут считаны в разыменованный указатель на указатель целочисленного типа *len* помещается

указатель *l*, а в разыменованный указатель на указатель на указатель символьного типа *s* помещается указатель на указатель символьного типа *sent*. Функция возвращает количество считанных предложений *i*.

3. Функция *delete_wrong_sentences()*

В функцию поступает указатель на указатель символьного типа *s*, указатель целочисленного типа *len* и целое число *i*. В целочисленную переменную *count*, в которой будет храниться количество предложений после обработки, помещается исходное количество предложений *i*. Далее с помощью двух циклов *for*, первый из которых проходит по массиву предложений *s*, а второй по каждой строчке *s[i]*, происходит проверка каждого символа и, если в предложении встречается цифра 7, то первый символ этого предложения становится \0, *count* уменьшается на 1, и с помощью оператора *break* завершаются циклы. Функция возвращает количество предложений после обработки *count*.

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.	https://pastebin.com/TX3JEtn2	https://pastebin.com/Dreh2FuW	Программа работает правильно

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 функции text. Обработка данных происходит с помощью функции delete_wrong_sentences.

Обработанные данные возвращаются пользователю на консоль, занятая память под нужды программы освобождается.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lab3.c

#include <stdio.h>

#include <stdlib.h>

```
int text(char*** s, int** len)
{
	char c;
	int i=0;
	int* l=*len;
	l=malloc(1*sizeof(int));
	char **sent=*s;
	sent=malloc((i+1)*sizeof(char*));
	while (c!='!')
	{
		scanf("%c",&c);
		if (c!=' ' && c!='\t' && c!='\n')
		{
			int number=0;
			int sent_size=1;
			sent[i]=malloc(sent_size*sizeof(char));
			*(sent[i]+number)=c;
			while (c!='.' && c!=';' && c!='?' && c!='!')
			{
				scanf("%c",&c);
				if (c!='\t' && c!='\n')
				{
					sent_size+=1;
					number=number+1;

sent[i]=realloc(sent[i],sent_size*sizeof(char));
					*(sent[i]+number)=c;
				}

			}
		sent[i]=realloc(sent[i],(sent_size+1)*sizeof(char));
```

```
                *(sent[i]+number+1)='\0';
                *(l+i)=sent_size;
                i=i+1;
                sent=realloc(sent,(i+1)*sizeof(char*));
                l=realloc(l,(i+1)*sizeof(int));
            }
            *len=l;
            *s=sent;
        }
        return i;
}

int delete_wrong_sentences(char** s,int* len, int i)
{
        int count=i;
        for (int t=0;t<i;t++)
        {
            for (int z=0;z<len[t];z++)
            {
                if (s[t][z]=='7')
                {
                    count--;
                    s[t][0]='\0';
                    break;
                }
            }
        }
        return count;
}

int main()
{
        int *len;
        char **s;
        int i=text(&s,&len);
        int count=delete_wrong_sentences(s,len,i);
        for (int t=0;t<i;t++)
        {
            if (s[t][0]!='\0')
```

```
            {
                printf("%s\n", s[t]);
            }
        }
        printf("%s", "Количество предложений до ");
        printf("%d", i-1);
        printf("%s", " и количество предложений после ");
        printf("%d\n", count-1);
        for (int j=0;j<i;j++)
        {
            free(s[j]);
        }
        free(s);
        return 0;
}
```