

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №3
по дисциплине «Программирование Си»
Тема: Использование указателей.

Студентка гр. 0382

Кривенцова Л.С.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Изучить как работают указатели и массивы в языке Си.

Задание.

Вариант 4.

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**

Основные теоретические положения.

Каждая переменная имеет своё место в оперативной памяти, т.е. **адрес**, по которому к ней обращается программа и может обращаться программист.

Унарная операция **&** даёт адрес объекта. Она применима только к переменным и элементам массива, конструкции вида **&(x-1)** и **&3** являются *незаконными*.

Формально, в языке Си нет специального типа данных для строк, но представление их довольно естественно - строки в языке Си это массивы символов, завершающиеся нулевым символом (**'\0'**). Это порождает следующие особенности, которые следует помнить:

- Нулевой символ является обязательным.
- Символы, расположенные в массиве после первого нулевого символа никак не интерпретируются и считаются мусором.
- Отсутствие нулевого символа может привести к выходу за границу массива.
- Фактический размер массива должен быть на единицу больше количества символов в строке (для хранения нулевого символа)
- Выполняя операции над строками, нужно учитывать размер массива, выделенный под хранение строки.

Строки могут быть инициализированы при объявлении.

char fgets(char *str, int num, FILE *stream)*

- Безопасный способ (явно указывается размер буфера)
- Считывает до символа переноса строки
- *Помещает символ переноса строки в строку-буфер (!)*

*int scanf(const char *format, arg1, arg2, ...argN);*

- **%s** в форматной строке для ввода строки
- Считывает символы до первого символа табуляции (не помещая его в строку)
- **Не контролирует размер буфера**
- **Потенциально опасна**

char gets(char* str);*

- **Не контролирует размер буфера**
- **Потенциально опасна**

Как вы уже могли догадаться, если строка в Си - массив символов, то массив строк это двумерный массив символов, где каждая строка - массив, хранящий очередную символьную строку.

Статический массив строк может быть также инициализирован при объявлении

Примеры:

```
// статический массив строк
char stat_strs[][LEN]={"Hello",
                        "It's string array",
                        "of three rows"
                        };

// создание динамического массива строк
char **dyn_strs = malloc(N * sizeof(char*));
for(i=0;i<N;i++){
    dyn_strs[i] = calloc(LEN, sizeof(char));
    strncpy(dyn_strs[i], stat_strs[i], LEN - 1); // копирование строк из stat_strs
}

// освобождение памяти
for(i=0;i<N;i++)
    free(dyn_strs[i]);
free(dyn_strs);
```

Выполнение работы.

Ход решения:

Используется стандартная библиотека языка си, её заголовочные файлы `stdio.h`, `stdlib.h`, `string.h` для работы со строками и `ctype.h` для работы с символами.

С помощью *malloc* выделяется память для двумерного динамического массива, куда будет записываться результат – преобразованный текст). На вход программы подаётся текст. Он считывается последовательно по предложениям с помощью функции *takephrase* (при считывании удаляя из начала предложения точнее не включая в новую строку, хранящую нынешнее предложение табуляции, пробелы и символы переноса строки). Затем с помощью условного оператора *if* и функции *suitable* проверяется, подходящее ли предложение (не содержит ли число 555), и в зависимости от

результата записывает (или нет) строку с предложением в динамический двумерный массив, который в конце выполнения программы является результатом её выполнения.

Далее ведётся подсчёт предложений (включенных и невключенных в итоговый текст) *notinclud++*; *lengthh++*; Совершается проверка, хватает ли места, выделенного для записи итогового текста, и если нет – выделяется новый объём памяти с помощью *realloc*. Сравнивается, если только что прочитанное предложение было конечной (заданной в условии) фразой, то цикл завершается.

В цикле производится печать массива (текста). Затем с помощью *free* освобождается выделенная под динамический массив память. Печатается последнее предложение с указанием длин исходного и итогового текстов.

Переменные:

1.Главной функции main():

int lengthh; - переменная– содержащая количество предложений.

int notinclud; - переменная – счётчик предложений исходного текста, не включенных в итоговый текст.

char phrase*; - адрес динамического массива, в который считывается строка (одно предложение исходного текста).

char end*; - адрес динамического массива, содержащий строку, которой заканчивается исходный текст (терминальное предложение).

int addmemory; - переменная, в которой содержится количество байт, которое следует выделить в памяти при её нехватке.

*char** result*; - адрес двумерного динамического массива, в который будет записываться отредактированный текст.

int i; - переменная-счётчик в цикле.

2.Функции takephrase():

int addmemory; - переменная, в которой содержится количество байт, которое следует выделить в памяти при её нехватке.

int lenphrase; - переменная, которая хранит индекс элемента динамического массива, в который записывать символ.

int sym; - переменная, в которую считывается символ.

*char *phrase;* - адрес динамического массива, в который считывается строка (одно предложение исходного текста).

3. Функции *int suitable(char* phrase)*:

int res; - переменная, значение которой возвращает функция. Логическая переменная, показывающая, подходит проверяемое предложение по условию или нет.

int i; - переменная счётчик для цикла.

Функции:

1. *main()*.

Функция осуществляет выделение памяти под динамические массивы, вызывает функцию считывания предложения *takephrase*; делает проверку, вызывая функцию проверки предложения *suitable*, записывает подходящие предложения в итоговый динамический массив (текст), в конце печатает его(в цикле) и освобождает память, печатает последнее предложение. Не принимает аргументов, возвращает 0 при корректной работе.

2. *takephrase()*.

Функция выполняет считывание предложения с клавиатуры. Выделяет память для массива в динамическом двумерном массиве. В цикле считывает символ, проверяя, не является ли он конечным символом предложения. Проверяет, хватает ли памяти для записи в массив, и если нет, выделяет с помощью *realloc*. Не принимает на вход аргументы.

3. *int suitable(char* phrase)*.

Функция выполняет проверку, подходит ли предложение (которое принимается на вход аргументом, *char* phrase*). Если в нём нет числа 555, то оно подходит и переменной *res* присваивается значение 1. В ином случае переменной *res* присваивается значение 0. Функция возвращает значение переменной *res*.

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.	<p>Nulla facilisi.</p> <p>Class aptenT taciti sociosqu ad litora torquent per cOnubia nostra,</p> <p>per inceptos himenaeos. 40 Nu555lla</p> <p>rutrum feugiat felis a pharetra.</p> <p>Sed finibus magna et mauris elementum tempus? Integer at quam et erat iaculis iaculis hendrerit a te4llus? Donec at nunc ac mauris suscipit venenatis.</p> <p>Sed finibus magna et mauris elementum tempus? Lorem ipsum</p> <p>dolor sit amet, consectetur adipiscing elit. Morbi</p> <p>co7ndimentum 555 ex justo, nec pharetra mauris vestibulum a.</p> <p>Suspendisse quis mi neque7.</p> <p>1 Vivamus eu nibh rhoncus, da456pibus ex non, sodales mi.</p> <p>Donec accumsan convallis ipsum</p> <p>vitae lacinia. Donec accumsan convallis ipsum vitae lacinia.</p> <p>Fusce finibus sapien magna, quis scelerisque ex sodales tristique. Nulla facilisi.</p> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>	<p>Nulla facilisi.</p> <p>Class aptenT taciti sociosqu ad litora torquent per cOnubia</p> <p>nostra, per inceptos himenaeos.</p> <p>40 Nu555lla rutrum feugiat felis a pharetra.</p> <p>Sed finibus magna et mauris elementum tempus? Integer at quam et erat iaculis iaculis hendrerit a te4llus?</p> <p>Donec at nunc ac mauris suscipit venenatis.</p> <p>Sed finibus magna et mauris elementum tempus? Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse quis mi neque7.</p> <p>1 Vivamus eu nibh rhoncus, da456pibus ex non, sodales mi.</p> <p>Donec accumsan convallis ipsum</p> <p>vitae lacinia.</p> <p>Donec accumsan convallis ipsum</p> <p>vitae lacinia.</p> <p>Fusce finibus sapien magna, quis scelerisque ex sodales tristique.</p> <p>Nulla facilisi.</p>	<p>Программа</p> <p>ВЫВОДИТ</p> <p>верный</p> <p>ответ.</p>
----	---	---	---

	Dragon flew away!	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Dragon flew away! Количество предложений до 16 и количество предложений после 15	
2.	kejrgerjbg;kirjgkejrg.erihgebrihg werivherb; kjerhvjhber srbdr;555 wergerger; 5555 ikbwejh bvejr bh; oweiurhgiuer 555.Dragon flew away!	kejrgerjbg; kirjgkejrg. erihgebrihg werivherb; kjerhvjhber srbdr; 5555 ikbwejh bvejr bh; Dragon flew away! Количество предложений до 7 и количество предложений после 5	Программа выводит верный ответ.

Выводы.

Были изучены возможности работы указателей и массивов в языке Си.

Разработана программа, выполняющая считывание исходных данных с клавиатуры в строки (динамические массивы). Для обработки введенного текста использовались указатели и адреса. Программа была разбита на отдельные функции.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb3.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
char* takephrase(){
    int addmemory = 100;
    int lenphrase = 0, sym;
    char *phrase = malloc(addmemory*sizeof(char));
    sym = getchar();
    if (sym=='\n');
    else { if (sym!=' ' && sym!='\t') phrase[lenphrase++] = sym;}
    sym = getchar();
    if (sym!=' ' && sym!='\t') phrase[lenphrase++] = sym;
    do {
        sym = getchar();
        phrase[lenphrase++] = sym;
        if (lenphrase == addmemory) {
            addmemory += 100;
            phrase = realloc(phrase, addmemory);
        }
    } while (sym != '.' && sym != ';' && sym != '?' && sym != '!');
    phrase[lenphrase] = '\0';
    return phrase;
}

int suitable(char* phrase){
    int res =1;
    for (int i = 0; i < strlen(phrase)-1; i++){
        if (phrase[i] == '5' && phrase[i+1] == '5' && phrase[i+2] == '5'
        && isalnum(phrase[i-1])==0 && isalnum(phrase[i+3]) == 0 )
            res = 0;
    }
    return res;
}

int main() {
    int lengthh = 0, notinclud = 0;
    char* phrase, *end = "Dragon flew away!";
    int addmemory = 100;
    char** result = malloc(addmemory*sizeof(char*));

    do { phrase = takephrase();
        if (suitable(phrase)!=0) {
            result[lengthh-notinclud] = phrase;
            lengthh ++;
        }
    }
```

```

        else {notinclud++;
              lengthh ++;}
        if ((lengthh - notinclud) == addmemory) {
            addmemory += 100;
            result= realloc(result, addmemory*sizeof(char*));
        }
    } while (strcmp(phrase, end));

    for (int i = 0; i < (lengthh - notinclud); i++){
        puts(result[i]);
        free(result[i]);
    }
    free(result);
    printf("Количество предложений до %d и количество предложений
после %d\n", lengthh -1 ,lengthh- notinclud -1);
    return 0;
}

```