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.

Студент гр. 0382 _____ Тюленев Т.В.

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 **readstr** (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 ***delete*** проверяется, подходящаее 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) *notlen++; len++;* 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 len; - переменная– содержащая количество предложений.

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

*char** str*;* - 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 *size*; - 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 readstr():

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

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

int sym; - переменная, в которую считывается символ.

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

3.Функции int delete(char* phrase):

int x; - переменная, значение которой возвращает функция. Логическая переменная, показывающая, подходит проверяющемое 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 ***readstr***; делает проверку, вызывая функцию проверки предложения ***delete***, 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. *readstr().*

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.*delete(char* str).*

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

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.	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 convallis ipsum vitae lacinia. Fusce finibus sapien magna, quis scelerisque ex sodales tristique. Nulla facilisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit.	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 vitae lacinia. Donec accumsan convallis ipsum vitae lacinia. Fusce finibus sapien magna, quis scelerisque ex sodales tristique. Nulla facilisi.	Программа выводит верный ответ.

	Dragon flew away!	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Dragon flew away! Количество предложений до 16 и количество предложений после 15	
2.	555. 655 565 556? 555; Dragon flew away!	655 565 556? Dragon flew away! Количество предложений до 3 и количество предложений после 1	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 введённого текста использовались указатели и адреса. Программа была разбита на отдельные функции.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.c

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

char* readstr();
int delete(char*);

int main() {
   int len = 0;
   int notlen = 0;
   char *str;
   char *end = "Dragon flew away!";
   int size = 2;
   char** result = malloc(size*sizeof(char*));

   do { str = readstr();
      if (delete(str)==0) {
         notlen++;
         len ++;
      }
      else {
           result[len-notlen] = str;
          len ++;
     }
      if ((len - notlen) == size) {
         size += 255;
         result= realloc(result, size*sizeof(char*));
      }
   } while (strcmp(str, end));

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

char* readstr(){
   int size = 255;
   int lenstr = 0, sym;
   char *str = malloc(size*sizeof(char));
   sym = getchar();
   if (sym=='\n');
   else { if (sym!=' ' && sym!='\t') str[lenstr++] = sym;}
```

```
    sym = getchar();
    if (sym!=' ' && sym!='\t') str[lenstr++] = sym;
    do {
        sym = getchar();
        str[lenstr++] = sym;
        if (lenstr == size) {
            size += 255;
            str = realloc(str, size);
        }
    } while (sym != '.' && sym != ';' && sym != '?' && sym != '!');
    str[lenstr] = '\0';
    return str;
}


int delete(char* str){
    int x =1;
 for (int i = 0; i < strlen(str)-1; i++){
     if (isalnum(str[i-1])==0 &&str[i] == '5' && str[i+1] == '5' && str[i+2] == '5' &&  isalnum(str[i+3]) == 0 )
         x = 0;
 }
    return x;
}
```