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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Программирование»

Тема: «Использование указателей»

Студент гр. 7381 ____ Лауцюс М.

Преподаватель ____ Берленко Т.А.

Санкт-Петербург

2017

Цель работы.

Приобретение навыков работы с указателями и динамической памятью в языке Си. Приобретение навыков работы с динамическими массивами, строками 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 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.

Заголовочные файлы стандартной библиотеки языка Си, используемые в данной лабораторной работе:

1. stdio.h - содержит определения макросов, константы и объявления функций и типов, используемых для различных операций стандартного ввода и вывода.

2. string.h- заголовочный файл для работы с Си-строками.

- char * strstr(const char * string1, const char * string2):

Описание:

Функция ищет первое вхождение подстроки string2 в строке string1. Возвращает указатель на первое вхождение строки string2 в строку string1, или пустой указатель, если строка string2 не является частью строки string1. В данном поиске нуль-символ не учитывается.

Параметры:

string1

Строка, в которой выполняется поиск.

string2

Подстрока для поиска в строке string1.

Возвращаемое значение:

Указатель на первое вхождение в string1 любой последовательности символов, указанных в string2.

Нулевой указатель, если последовательность символов строки string2 не входит в string1.

3. stdlib.h- содержит функции для преобразования чисел в текст, выделения памяти, генерации случайных чисел и др. функций-утилит.

- void * malloc(size_t sizemem):

Описание:

Функция malloc выделяет блок памяти, размером “sizemem” байт, и возвращает указатель на начало блока.

Содержание выделенного блока памяти не инициализируется, оно остается с неопределенными значениями

Параметры:

sizemem

Размер выделяемого блока памяти в байтах.

Возвращаемое значение:

Указатель на выделенный блок памяти. Тип данных на который ссылается указатель всегда void*, поэтому это тип данных может быть приведен к желаемому типу данных.

Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель.

- void free(void * ptrmem):

Описание:

Функция free освобождает место в памяти.]

Параметры:

ptrmem

Указатель на блок памяти, ранее выделенный функциями malloc, calloc или realloc, которую необходимо высвободить. Если в качестве аргумента передается нулевой указатель, никаких действий не происходит.

- void * realloc(void * ptrmem, size_t size):

Описание:

Функция realloc выполняет перераспределение блоков памяти. Размер блока памяти, на который ссылается параметр ptrmem изменяется на size байтов. Блок памяти может уменьшаться или увеличиваться в размере.

Эта функция может перемещать блок памяти на новое место, в этом случае функция возвращает указатель на новое место в памяти. Содержание блока памяти сохраняется даже если новый блок имеет меньший размер, чем старый. Отбрасываются только те данные, которые не вместились в новый блок. Если новое значение size больше старого, то содержимое вновь выделенной памяти будет неопределенным.

В случае, если ptrmem равен NULL, функция ведет себя именно так, как функция malloc, т. е. выделяет память и возвращает указатель на этот участок памяти.

В случае, если size равен 0, ранее выделенная память будет освобождена, как если бы была вызвана функция free, и возвращается нулевой указатель.

Параметры:

ptrmem

Указатель на блок ранее выделенной памяти
функциями malloc, calloc или realloc для перемещения в новое место. Если этот параметр — NULL, просто выделяется новый блок, и функция возвращает на него указатель.

size

Новый размер, в байтах, выделяемого блока памяти. Если size равно 0, ранее выделенная память освобождается и функция возвращает нулевой

указатель, ptrmem устанавливается в 0.

Возвращаемое значение:

Указатель на выделенный блок памяти. Тип данных на который ссылается указатель всегда void*, поэтому это тип данных может быть приведен к желаемому типу данных.

Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель.

Возвращаемое значение:

Указатель на перераспределенный блок памяти, который может быть либо таким же, как аргумент ptrmem или ссылаться на новое место. Тип данных возвращаемого значения всегда void*, который может быть приведен к любому другому. Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель, и блок памяти, на который указывает аргумент ptr остается неизменным.

4. ctype.h - содержит прототипы функций для обработки символов.

int isalpha(int character)

Функция возвращает истинное значение true, если её аргумент - буква, и false(ложь) в других случаях

int isdigit(int character)

Функция возвращает истинное значение true, если её аргумент - десятичная цифра, и false(ложь) в других случаях.

Вывод:

В ходе выполнения лабораторной работы было произведено ознакомление с указателями и динамической памятью. Была создана программа для обработки динамического массива строк.

Исходный код проекта:

Файл Makefile:

```
all: Dragon.c
    gcc –o Dragon Dragon.c
```

Файл Dragon.c (1 вариант):

```
#define STR "Dragon flew away!"
#define CH "555"
#define SIZE 50
#define PLUSSIZE 50
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
int main() {
    int n=0, i=0, m=0, k=0;
    char c;
    do{
        i=0;
        char* string =(char*)calloc(SIZE, sizeof(char));
        do{
            do{
                c = getc(stdin);
            }while((c==' '||c=='\t')&&!i);
            string[i++]=c;
            //i++;
            if(!((i-SIZE+1)%PLUSSIZE))
                string=(char*)realloc(string, i+PLUSSIZE);
}while(c!='.'&&c!=';'&&c!='?'&&!strstr(string,STR));
        n++;
        string[i]='\0';
        char* a=strstr(string,CH);
if(!(a&&!isalpha(*(a-1))&&!isalpha(*(a+3))&&!isdigit(*(a-
1))&&!isdigit(*(a+3)))){
            printf("%s\n", string);
            m++;
```

```
        }
        if(strstr(string,STR))
            k=1;
        free(string);
    }while(!k);
    printf("Количество предложений до %d и количество
предложений после %d\n", n-1, m-1);
    return 0;
}
```

Файл Dragon.c (2 вариант):

```
#define STR "Dragon flew away!\0"
#define SIZE 50
#define PLUSSIZE 50
#include <stdio.h>
#include <string.h>
#include  <stdlib.h>
#include <ctype.h>
char* read_string(){
    char c='a';
    int i=0,size=SIZE;
    char* string=(char*)calloc(SIZE, sizeof(char));
    while(c!=';'&&c!='.'&&c!='?'&&strstr(string,STR)==NULL
){
         c=getc(stdin);
         string[i]=c;
         i++;
         if(i==size){
              string=(char*)realloc(string,
size+PLUSSIZE);
              size=size+PLUSSIZE;
         }
    }
    string=(char*)realloc(string, ++size);
    string[i]='\0';
```

```
        return string;
}
int delete_555(char** pointer,int n){
     int i;int j;
     int m=n;
     for(i=0;i<m;i++){
         pointer[i];
         char* a=strstr(pointer[i],"555");
if(a&&!isalpha(*(a-1))&&!isalpha(*(a+3))&&!isdigit(*(a-
1))&&!isdigit(*(a+3))){
             m--;
             free(pointer[i]);
             for(j=i;j<m;j++){
                 pointer[j]=pointer[j+1];
             }
             i--;
         }
     }
     return m;
}
void delete_(char** pointer,int m){
    int i, j, k;
    for(i=0;i<m;i++){
        for(j=0;pointer[i][j]==' '||pointer[i][j]=='\t';){
               for(k=j;pointer[i][k]!='\0';k++){
                    pointer[i][k]=pointer[i][k+1];
               }
            }
    }
}
int main() {
    int i=0, n=0, m=0;char c;
    char** pointer=(char**)malloc(sizeof(char*));
    char* string=(char*)malloc(sizeof(char));
    while(strstr(string,STR)==NULL){
    string=read_string();
```

```
    n++;
    pointer=(char**)realloc(pointer, n*sizeof(char*));
    pointer[n-1]=string;
    i++;
    }
    m=delete_555(pointer, n);
    delete_(pointer, m);
    for(i=0;i<m;i++)
        printf("%s\n", pointer[i]);
    printf("Количество предложений до %d и количество
предложений после %d",n-1,m-1);
    for(i=0;i<m;i++)free(pointer[i]);
    free(pointer);
    return 0;
}
```