МИНОБРНАУКИ РОССИИ

Санкт-Петербургский государственный

электротехнический университет

«ЛЭТИ» им. В.И. Ульянова (Ленина)

Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Программирование»

Тема: Использование указателей

Студент(ка) гр. 7381 Алясова А.Н.

Преподаватель Берленко Т.А.

Санкт-Петербург

2017

Цель работы:

Познакомиться с указателями, строками, динамической памятью, а также с функциями для работы с ними.

Задание:

Написать 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 цифры внутри слов, должны быть удалены (это не касается слов, которые начинаются/заканчиваются цифрами).
- 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, необходимые для создания проекта:

1. <stdio.h> - содержит прототип функции «int printf(const char* format [,

argument]...);», которая используется для вывода в поток вывода.

Синтаксис:

#include < stdio.h >

*int printf (const char *format, ...);*

Аргументы:

format – указатель на строку с описанием формата.

Возвращаемое значение:

При успешном завершении вывода возвращается количество выведенных символов.

При ошибке возвращается отрицательное число.

2. <stdlib.h> - содержит прототипы функций «void* calloc (size_t num, size_t size);» и «void free (void* ptr);», которые динамически выделяют память под массив данных, предварительно инициализируя её нулями и высвобождают динамически выделенную ранее память.

Синтаксис:

#include < stdlib.h >

*void * calloc(size_t number, size_t size);*

Аргументы:

number -количество элементов массива, под который выделяется память.

size - размер одного элемента 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 типом данных. Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель.

Описание:

Функция **calloc** выделяет блок памяти для массива размером — **num** элементов, каждый из которых занимает **size** байт, и инициализирует все свои биты в нулями. В результате выделяется блок памяти размером **number * size** байт, причём весь блок заполнен нулями.

Синтаксис:

#include < stdlib.h >

*void free(void * ptrmem);*

Аргументы:

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.

Возвращаемое значение:

Нет.

Описание:

Функция **free** освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова **malloc**, **calloc** или **realloc** освобождается. То есть освобожденная память может дальше использоваться программами или ОС.

3. <ctype.h> - содержит прототипы функций «int isdigit(int character);» и «int isspace(int character);», которые возвращают истинное значение true, если аргумент - десятичная цифра, и false(ложь) в других случаях; и возвращают истинное значение true, если аргумент - любой знак пробела, и false(ложь) в других случаях.

Синтаксис:

#include < ctype.h >

int isdigit(int character);

Аргументы:

character -символ для проверки, передается в функцию как значение типа int, или EOF.

Возвращаемое значение:

Значение, отличное от нуля (т.е. истинно), если аргумент функции — это десятичная цифра . Ноль (т.е. ложь), в противном случае.

Описание:

Функция **isdigit** проверяет аргумент, передаваемый через параметр **character**, является ли он десятичной цифрой.

Синтаксис:

#include < ctype.h >

int isspace(int character);

Аргументы:

character - символ для проверки, передаётся в функцию как значение типа int, или EOF.

Возвращаемое значение:

Значение, отличное от нуля (т.е. истинно), если аргумент функции — это символ пробела. Ноль (т.е. ложь), в противном случае.

Описание:

Функция **isspace** проверяет параметр **character**, является ли он символом пробела.

Вывод:

В результате работы были освоены указатели, функции для работы с динамической памятью malloc, calloc, realloc и free, а также некоторые функции для работы со строками и отдельными символами.

Исходный код проекта:

Makefile

```
all: main.o
	gcc main.o
main.o: main.c
gcc -c main.c
```

main.c

Makefile

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

int main() {
    char c;
    char *text = calloc(50, sizeof(char));
    int text_size = 0;
    while ((c = getchar()) != '!') {
        if (c=='\n' || c=='\t')
            continue;
        if (text_size%50==0 && text_size>0)
            text = realloc(text, (text_size+50)*sizeof(char));
        text[text_size] = c;
        text_size++;
                                }
    if (text_size%50==0)
            text = realloc(text, (text_size+50)*sizeof(char));
    text[text_size]='!';
    text_size++;


    int i, k, proverka, n=0, m=0, nachalo=0, konetc;

    while (1){
 //бесконечный цикл, если истинна
        for (k=nachalo;;k++){
            if (text[k]==';' || text[k]=='.' || text[k]=='?' || text[k]=='!'){
                konetc = k;
                n++;
                break;
                                                        }
                        }
 // нашла границы предложения, количество предложений изначально

        for (k=nachalo+1, proverka = 1; k<=konetc; k++){
            if ( isdigit(text[k]) && !isspace(text[k-1]) && !isspace(text[k+1]) &&
!isdigit(text[k-1]) && text[k+1]!=';' && text[k+1]!='.' && text[k+1]!='!' &&
text[k+1]!='?' )
                                        {
//isdigit - возвращает ненулевое значение, если десятичное число
//isspace - возвращает true, если пробельный символ
```

```
  while (isdigit(text[k]))
                k++;
             if (!isspace(text[k]) && text[k]!='!' && text[k]!=';' && text[k]!='?' &&
text[k]!='.')
             {
                proverka = 0;
                break;
             }
                                      }
 }
 // нашла границы предложения, количество предложений изначально

       for (k=nachalo+1, proverka = 1; k<=konetc; k++){
          if ( isdigit(text[k]) && !isspace(text[k-1]) && !isspace(text[k+1]) &&
!isdigit(text[k-1]) && text[k+1]!=';' && text[k+1]!='.' && text[k+1]!='!' &&
text[k+1]!='?' )
                                      {
//isdigit - возвращает ненулевое значение, если десятичное число
//isspace - возвращает true, если пробельный символ

  while (isdigit(text[k]))
                k++;
             if (!isspace(text[k]) && text[k]!='!' && text[k]!=';' && text[k]!='?' &&
text[k]!='.')
             {
                proverka = 0;
                break;
             }
                                      }
                                      }
// проверела, есть ли цифры в словах этого предложения

       if (proverka){
          for (k=nachalo; k<=konetc; k++)
             printf("%c", text[k]);
          printf("\n");
          m++;
                }
// если цифр нет, вывела предложение и перенесла строку, увеличила
предложения после

       if (text[konetc]=='!')
          break;
// если это последнее предложение, прерываю цикл
```

```
        nachalo=konetc+1;
        while (1)
            if (isspace(text[nachalo]))
                nachalo++;
            else
                break;
// перешла к следующему предложению
    }
    n--;
    m--;
    printf("Количество предложений до %d и количество предложений после
%d",n,m);
 free (text);
    return 0;
}
```