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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Программирование»
Тема: Указатели и массивы

Студент гр. 7381 _____ Смирнов М.А.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2017

Цель работы.

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:

1. Каждое предложение должно начинаться с новой строки.
2. Табуляция в начале предложения должна быть удалена.
3. Все предложения, в которых есть цифры внутри слов, должны быть удалены (это не касается слов, которые начинаются/заканчиваются цифрами).
4. 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. – содержит прототипы функций для выделения памяти “void * malloc(size_t sizemem);” “void * realloc(void * ptrmem, size_t size);” и “void free(void * ptrmem);” (a) void * malloc(size_t sizemem);

***Описание**:* Функция *malloc* выделяет блок памяти и возвращает указатель на начало блока. Содержание выделенного блока памяти не инициализируется, оно остается с неопределенными значениями.

Параметры: 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 * 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 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.

*void free(void * ptrmem);*

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. Обратите внимание, что эта функция оставляет значение *ptr* неизменным, следовательно, он по-прежнему указывает на тот же блок памяти, а не на нулевой указатель. 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.

Возвращаемое значение: Функция не имеет возвращаемое значение.

2. – содержит прототипы функций "*int printf(const char* format [, argument]...)*" и "int *getchar (void);*", которые используются для ввода из потока ввода и вывода в поток вывода.

3. – содержит прототип функции “*size_t strlen(const char * string)*; *“ a. size_t strlen(const char * string)*;

Описание: Длина Си-строки определяется по достижению нулевого символа — нуль терминатор. Функция *strlen* видит начало Си-строки и начинает сначала считать количество символов (байтов, отводимых под каждый символ), этот процесс выполняется до тех пор, пока не будет достигнут завершающий нулевой символ. Обратите внимание на то, что завершающий

нулевой символ не входит в длину строки. Он является служебным символом, для обозначения завершения Си-строки.

Параметры: *string* Си-строка.

Возвращаемое значение: Длина строки.

4. – содержит прототипы функций "*int isspace(int character);*" и "*int isdigit(int character);*" a. *int isdigit(int character*);

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

Параметры: *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. b. *int isspace(int character)*; Описание: Функция isspace проверяет параметр *character*, является ли он символом пробела. Обратите внимание на то,что символ пробела — это, на самом деле, несколько символов.

Параметры: *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 символ пробела. Ноль (т.е. ложь), в противном случае.

Функция main.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iso646.h>
#include <ctype.h>


int prove(char* s, int i);

int main()
{
    char* text; char c = 'x'; int i=0; int len=100;
    char* rezt; char* tp;char* ltext;

    text=(char*)malloc(len*sizeof(char));

while(c!= '!' ){
c=getchar();
text[i] = c;
i++;
if(i == len){
len = len +len;
text = (char*)realloc(text, len*sizeof(char));
}
}

text[i+1]='\0';

int fire=0; int che;
for(che=0;che<strlen(text);che++)
{
    if (text[che]==';' || text[che]=='.' || text[che]=='?')
        fire=fire+1;
}
int pen;


for(pen=0;pen<strlen(text);pen++)
{
    if (text[pen]=='\t')
        text[pen]=' ';
}
ltext=(char*)malloc(strlen(text)*sizeof(char));
int j=0;
int k;
for (k=0; k<strlen(text);k++){
if (text[k]==' ')
{
    if (j==0) continue;
    if (text[k+1]==' ') continue;
}
    ltext[j]=text[k];
    j++;
}

int hel;
//перенос строки

for(hel=0;hel<strlen(ltext);hel++)
{
   if (ltext[hel]==';' || ltext[hel]=='.' || ltext[hel]=='!' || ltext[hel]=='?')
    ltext[hel+1] = '\n';


}

int a=0, b, f=0, g=0;
rezt=(char*)malloc(strlen(ltext)*sizeof(char));
    while(ltext[a]!='!')
    {
    b=0;
    tp=(char*)malloc(len*sizeof(char));
    do {
        tp[b]=ltext[a];
```

```
      }while(ltext[a]!='\n'&&ltext[a]!='!');

        if (prove(tp, b)==1)
                for(f=0;f<b;g++,f++)
                rezt[g]=tp[f];
    free(tp);
   }

rezt[g]='!';

        if (rezt[0]=='\n')
        {
            for(a=1;rezt[a]!='!';a++)
            printf("%c",rezt[a]);
        }
           else
            if (rezt[0]==' ')
            {
            for(a=1;rezt[a]!='!';a++)
            printf("%c",rezt[a]);
            }
               else for(a=0;rezt[a]!='!';a++)
                printf("%c",rezt[a]);

printf("!\n");

int mm=0; int ff;
for(ff=0;rezt[ff]!='!';ff++)
{
    if (rezt[ff]==';' || rezt[ff]=='.' || rezt[ff]=='?')
        mm=mm+1;
}
printf("Количество предложений до %d и количество предложений после %d\n",(fire),(mm));
free(text);
free(ltext);
free(rezt);
 return 0;
```

```
int mm=0; int ff;
for(ff=0;rezt[ff]!='!';ff++)
{
    if (rezt[ff]==';' || rezt[ff]=='.' || rezt[ff]=='?')
        mm=mm+1;
}
printf("Количество предложений до %d и количество предложений после %d\n",(fire),(mm));
free(text);
free(ltext);
free(rezt);
 return 0;
}


int prove(char* s, int i)
{
    int a;char sw[i]; int b;int c;int d;
    for (a=0;a<i;a++)
    {
        d=a;
        if (d==' ') d++;
        b=0;
        while(isalnum(s[d]))
        {sw[b]=s[d];
        b++;
        d++;
        }
        for(c=1;c<b;c++)
        {if(isdigit(sw[0])||isdigit(sw[b-1]))continue;
            if (isdigit(sw[c]) && isalnum(sw[c-1]) && isalnum(sw[c+1]))
            return 0;
        }

    }
   return 1;
}
```

Вывод: В процессе выполнения лабораторной работы была написана программа, позволяющая редактировать текст: удалять табуляцию, удалять предложения с цифрами в середине слова, а также переводить каждое новое предложение на новую строку.
В ходе работы были получены необходимые знания об указателях и массивах языка СИ.