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

Студентка гр. 7381 Кушкоева А.О.

Преподаватель Берленко Т.А.

Санкт-Петербург

2017

Цель

Освоить указатели, строки, динамическую память и функции 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, которые заканчиваются на '?' 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.

- Указатели

Указатель – это переменная, содержащая адрес другой переменной.
Синтаксис объявления указателя:

```
<тип_переменной_на_которую_ссылается_указатель>* <название переменной>;
```

- Динамическая память

Для работы с динамической памятью используются следующие функции:

- malloc (void* malloc (size_t size)) - выделяет блок из size байт и возвращает указатель на начало этого блока
- calloc (void* calloc (size_t num, size_t size)) - выделяет блок для num элементов, каждый из которых занимает size байт и инициализирует все биты выделеного блока нулями
- realloc (void* realloc (void* ptr, size_t size)) - изменяет размер ранее выделенной области памяти на которую ссылается указатель ptr. Возвращает указатель на область памяти, измененного размера.
- free (void free (void* ptr)) - высвобождает выделенную ранее память.

- Строки

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:

- 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

Вывод

В ходе лабораторной работы были освоены указатели, строки, динамическую память и функции для работы с ними.

Исходный код

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
int h=0;
int i=1;
int m=0;
int k=0;//количество предложений до обработки
int p=0;//после обработки
char* text=malloc(i*sizeof(char));
char c=getchar();
while(c!='!')
{
text[i-1]=c;
c=getchar();
text=(char*)realloc(text, (++i) *sizeof(char));
}
text[i-1]='!';
text[i]='\0';
char* text2=malloc(1 * sizeof(char));
i=0;
while(text[i]!='!')
{
	while(text[i]=='\t' || text[i]==' ')
	i++;

	while(text[i]!='.' && text[i]!=';' && text[i]!='?'  && text[i]!='!')
	{
	text2[m]=text[i];
	text2=realloc(text2, ((++m)+1) * sizeof(char));
	i++;
	    ++h;
	}

	if(text[i]=='.' || text[i]==';')
	    {
	text2[m]=text[i];
```

```
        text2=realloc(text2, ((++m)+1) * sizeof(char));
        i++;
        text2[m]='\n';
        text2=realloc(text2, ((++m)+1) * sizeof(char));
        k++;
        p++;
        h=0;
        }

        if(text[i]=='?')
        {
        k++;
        i++;
        m=m-h;
        h=0;
        }
}
text2[m]='!';
text2=realloc(text2, ((++m)+1) * sizeof(char));
text2[m]='\0';
printf("%s\n", text2);
printf("Количество предложений до %d и количество предложений после %d",
k, p);
    free(text);
    free(text2);
return 0;
}
```