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 гр. 7381 _____ Павлов А.П.

Преподаватель _____ Берленко Т.А.

Санкт-Петербург
2017

Цель работы.

Изучить указатели в языке Си и научиться основам работы с ними. Научиться работать с динамической памятью в языке Си. Освоить работу со строками.

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.

Заголовочные файлы, необходимые для создания проекта:
1. 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 -оличество 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

Вывод

В ходы выполнения лабораторной работы были изучены понятие указателя, синтаксис его объявления, а также использование. Изучены понятия динамической памяти, функции для работы с ней в C (выделение через malloc, calloc, realloc и освобождение через free). Изучено представление строк в C, а так же методы работы с ними.

Исходный код проекта

Файл main.c

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

#define N 15

int main(){
   int i=0, k=0, per=0,current=0, l=0, m;
   char c;
   char *str1=(char*)calloc(N, sizeof(char));
   while((c=getchar())!='!')
   {   if(c=='\t'||c=='\n')
         continue;
      if(i%N==0 && i>0)
         str1=(char*)realloc(str1, (i+N+2)*sizeof(char));
      str1[i]=c;
      if(str1[i]=='.'||str1[i]==';'||str1[i]=='?')
         k++;
      i++;
   }
   str1[i++]='!';
   str1[i]='\0';
   i=0;
   for(m=0;m<=k;m++)
   {
      for(current=i;str1[i]!='.' && str1[i]!=';' && str1[i]!='!' && str1[i]!='?';i++);
      if(str1[i]!='?')
      {
         per=i;
         for(i=current;str1[i]==' ';i++);
         for(;i<=per;i++)
           printf("%c", str1[i]);
         printf("\n");
         l++;
      }
      i++;
```

```
    }
    printf("Количество предложений до %d и количество предложений после
%d\n", k, l-1);
    free(str1);
    return 0;
    }
```

Файл Makefile

```
All: main.o
        gcc main.o

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