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 _____ Трушников А.П.

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).

Основные теоретические положения.

Заголовочные файлы, необходимые для создания проекта:

<stdlib.h>– содержит прототипы функций для выделения памяти “void * malloc(size_t sizemem);” “void * realloc(void * ptrmem, size_t size);” и “void free(void * ptrmem);”

1. 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.

2. 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.

3. void free(void * ptrmem);

Описание:

Функция free освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова malloc, calloc или reallocосвобождается. 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.

<stdio.h> содержит прототипы функций "int printf(const char* format [, argument]...);" и "int scanf(const char* format [, argument]...);", которые используются для ввода из потока ввода и вывода в поток вывода.

Вывод.

В процессе 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>

int check_word(char*word, int len) {
  int i;
  int l=0, r=len-1;
  while (word[l] >= '0' && word[l] <= '9' && l < len) {
        l++;
  }
  if (l >= len-1)
        return 1;
  while (word[r] >= '0' && word[r] <= '9' && r >= 0) {
        r--;
```

```
  }
  if (r <= 0)
        return 1;
  for (i = l; i <= r; i++) {
        if (word[i] >= '0' && word[i] <= '9') {
              return 0;
        }
  }
  return 1;
}

int main() {
   int before=0;
  char*str = (char*)malloc(100*sizeof(char));
  char**text = NULL;
  int i,l, line,linepos;
  int w = 0;
  int end = 0;
  int res = 0;

  line = 0;
  linepos = 0;
  while ((res = scanf("%[^ .;?\n\t]",str)) != EOF) {
        l = 0;
        if (res != 0) {
              while (str[l] != '\0') {
                    l++;
              }
              w=check_word(str, l);
              if (w == 0) {
```

```
                scanf("%[^.;?]", str);
                scanf("%c", str);
                before++;
                linepos = 0;
                continue;
        }
        if (linepos == 0) {
                text = realloc(text, (line + 1) * sizeof(char*));
                text[line] = NULL;
        }
        text[line] = realloc(text[line], (linepos + l) * sizeof(char));
        for (i = 0; i < l; i++) {
                text[line][linepos+i] = str[i];
        }
        linepos += l;
}
scanf("%c", str);
switch (str[0]) {
case '!':
        end = 1;
        break;
case '\t':
case ' ':
        if (linepos != 0) {
text[line] = realloc(text[line], (linepos + 1) * sizeof(char));
                text[line][linepos] = str[0];
                linepos++;
        }
        break;
case '.':
```

```
        case ';':
        case '?':
                text[line] = realloc(text[line], (linepos + 2) * sizeof(char));
                text[line][linepos] = str[0];
                text[line][linepos+1] = '\0';
                line++;
                linepos = 0;
                before++;
                break;
        case '\n':
                if (linepos != 0) {
                        text[line] = realloc(text[line], (linepos + 1) * sizeof(char));
                        text[line][linepos] = ' ';
                        linepos++;
                }
                        break;
        default:
                if (linepos != 0) {
                        text[line] = realloc(text[line], (linepos + 1) * sizeof(char));
                        text[line][linepos] = str[0];
                        linepos++;
                        break;
                }
        }
}

for (i = 0; i < line; i++) {
        printf("%s\n", text[i]);
}
printf("Dragon flew away!\n");
```

```
    printf("Количество предложений до %i и количество предложений после %i\n",before,line);
    free(str);
    for(i=0;i<line;i++){
        free(text[line]);
    }
    free(text);
  return 0;
}
```

Фаил **“Makefile”**

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