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 Адамов Я.В.

Преподаватель Берленко Т.А.

Санкт-Петербург

2017

Лабораторная работа №3

Использование указателей

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 стандартной библиотеки языка С, необходимые для выполнения данной лабораторной работы: stdio.h, stdlib.h, string.h, ctype.h.

- Библиотека stdio.h содержит прототипы функций стандартного ввода и вывода.

- Библиотека stdlib.h содержит прототипы функций для динамического выделения памяти:

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

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.

Параметры:

- 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.

3) **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.

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.В случае, если size равен 0, ранее выделенная память будет освобождена, как если бы была вызвана функция free, и возвращается нулевой указатель.

4) **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.

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.

- Библиотека string.h содержит прототип функции strlen:

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.

- Библиотека ctype.h содержит прототипы следующих функций:

1) **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.

2) **int isspace(int character);**

Описание: функция isspace проверяет параметр character, является ли он символом пробела (пробела, табуляции, переноса строки).

Параметры:

- 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.

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 программы:

menu.c:

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


int main()
{
   int i=0, d=1;
   char c;
   char *text1=malloc(101*sizeof(char));
   while ((c=getchar()) != '!')
   {
      text1[i++]=c;
      if ((i%100)==0)
      {
         text1=realloc(text1,100*sizeof(char)*(++d)+sizeof(char));
      }
   }
   text1[i++]='!';
   text1[i]='\0';
   i=0;


   int bil_probel, t, b, n=0, m=0;
   char *text2 = malloc(strlen(text1)*sizeof(char));
   char *text2_start = text2;
   while(text1[i]!='\0')
   {
      n++;
      b = t = bil_probel = 0;
      while ((text1[i]!=';') && (text1[i]!='.') && (text1[i] != '?') && (text1[i] != '!'))
      {
         if ((!isspace(text1[i])) || (bil_probel))
            {
               text2[t++]=text1[i];
               if (!bil_probel)
                  bil_probel = 1;
            }
         i++;
      }
      text2[t++]=text1[i++];
      text2[t]='\0';
      for(t = 1; t < strlen(text2); t++)
         if (isdigit(text2[t]) && !isdigit(text2[t-1]) && !isspace(text2[t-1]) && !isspace(text2[t+1]) &&
!(text2[t+1]=='!') && !(text2[t+1]==';') && !(text2[t+1]=='.') && !(text2[t+1]=='?'))
         {
```

```
            while(isdigit(text2[t]))
               t++;
            if (!isspace(text2[t]) && !(text2[t]=='!') && !(text2[t]==';') && !(text2[t]=='.') && !(text2[t]=='?'))
               {
                  b=1;
                  break;
               }
         }

      if (!b)
      {
         printf("%s\n",text2);
         m++;
      }

      text2 = text2_start;
   }
   printf("Количество предложений до %d и количество предложений после %d\n",n-1,m-1);
   free(text1);
   free(text2);
   return 0;
}
```

Makefile

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