МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: Работа со строками в языке С

Студент гр. 9303 _____ Максимов Е.А.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2019

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Максимов Е.А.

Группа 9303

Тема работы: Работа со строками в языке С

Исходные данные:
Набор предложений, разделённых точкой, предложения - набор слов, разделённых пробелом или запятой, слова - набор латинских букв и цифр.

Содержание пояснительной записки:
«Аннотация», «Содержание», «Введение», «Описание кода программы», «Примеры работы программы», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
22 страницы

Дата выдачи задания: 15.10.2019

Дата сдачи реферата: 08.12.2019

Дата защиты реферата: 09.12.2019

Студент _____ Максимов Е.А

Преподаватель _____ Чайка К.В.

АННОТАЦИЯ

В данной работе была создана программа на языке программирования C, которая производит обработку текста посредством набора функций. Программе на вход подаётся текст. Текст представляет собой предложения, разделённые точкой, предложения - набор слов, разделённые пробелом или запятой, слова - набор латинских букв и цифр. Программа сохраняет текст в динамический массив строк и оперирует далее только с ним. Программа удаляет все повторно встречающиеся предложения (без учёта регистра).

Далее, программа запрашивает у пользователя одно из следующих доступных действий:

0) Выйти из программы.
1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку, длина которой больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является изучение функций стандартной библиотеки языка С и разработка программы, которая редактирует текст по заранее заданным правилам.

Для достижения поставленной цели требуется решить следующие задачи:

1) изучение основ языка программирования С для реализации данной программы;

2) разработка программного кода;

3) сборка программы из полученного сегментированного программного кода;

4) тестирование программного кода.

1. ОПИСАНИЕ КОДА ПРОГРАММЫ

1.1. Функция чтения введённого текста

```
char** readText(int *sentence_count){
    char **text = (char**)calloc(1, sizeof(char*));
    if (text == NULL)
        return NULL;
    int i=0;
    int sentence_max=0;
    char c = getchar();
    do{
        i=0;
        if((text=(char**)realloc(text,
((*sentence_count)+1)*sizeof(char**)))==NULL)
            return NULL;
        do{
              if
((text[*sentence_count]=(char*)realloc(text[*sentence_count],
sizeof(char*)*(i+3)))==NULL)
                    return NULL;
            text[*sentence_count][i]=c;
            i++;
        }while((c=getchar())!='.');
        text[*sentence_count][i]='.';
        text[*sentence_count][i+1]='\0';
        (*sentence_count)++;
         if (i+2>sentence_max)
              sentence_max=i+2;
    }while((c = getchar())!='\n');

    char checker[*sentence_count][sentence_max];
    int j;
    for(i=0; i<*sentence_count; i++){
         strcpy(checker[i], text[i]);
         for(j=0; checker[i][j]; j++)
             checker[i][j]=tolower(checker[i][j]);
    }
    for(i=0; i<*sentence_count; i++)
         for(j=i+1; j<*sentence_count; j++)
              if(!strcmp(checker[i], checker[j])){
                  free(text[j]);
                    (*sentence_count)--;
                    for(int k=j; k<*sentence_count; k++){
                         text[k]=text[k+1];
                         strcpy(checker[k], checker[k+1]);
                    }
                    j--;
              }
    return text;
}
```

Принимает на вход указатель на целое значение количества предложений (изначально, равное 0). Программа получает данные из основного потока ввода.

Функция записывает полученные предложения в динамический массив. Далее, функция удаляет одинаковые предложения без учёта регистра.

Возвращает указатель на динамический массив строк. В ходе работы функции изменяется значение количества предложений.

1.2. Функция проверки на гласную букву

```
int isVowel(char ch){
    const char vowels[] = "aeiouyAEIOUY";
    if (strchr(vowels, ch)==NULL)
         return 0;
    return 1;
}
```

Принимает на вход символ. Возвращает:

• 0, если символ не является гласной буквой;

• 1, если является гласной буквой.

1.3. Функция замены гласных букв

```
void changeVowel(char **text, int sentence_count){
    for(int i=0; i<sentence_count; i++)
         for(int j=0; j<strlen(text[i]); j++)
              if(isVowel(text[i][j]))
                   text[i][j]++;
    return;
}
```

Принимает на вход указатель на динамический массив строк и значение количества предложений. Изменяет в каждой строке символ гласной буквы на следующую букву в алфавите.

1.4. Функция проверки предложения по маске

```
int isToBe(char* sentence){
    char* ptr1 = strstr(sentence, "To ");
    char* ptr2 = strstr(sentence, " or not to ");
    int len1, len2;
    if(ptr1==sentence && ptr2!=NULL){
         len1 = ptr2-ptr1-3;
         len2 = sentence+strlen(sentence)-(ptr2+12);
         if(len1>len2)
              return -len1;
         else return len2;
    }
    else return 0;
}
```

Принимает на вход указатель строку. Проверяет строку на тип конструкции “To <подстрока1> or not to <подстрока2>”. Возвращает:

• 0, если строка не является конструкцией данного типа;

• целое значение меньше 0, если длина подстроки №1 больше, чем длина подстроки №2;

• целое значение больше 0, если длина подстроки №2 больше, чем длина подстроки №1.

1.5. Функция поиска длиннейшей подстроки из маски

```
void findToBe(char **text, int *sentence_count){
    int length=0;
    char *ptr1 = (char*)calloc(1, sizeof(char));
    char *ptr2 = (char*)calloc(1, sizeof(char));
    for(int i=0; i<*sentence_count; i++){
         length = isToBe(text[i]);
         if(length<0){
              length = -length;
              ptr1 = (char*)realloc(ptr1, length+1);
              for(int j=0; j<length; j++)
                   ptr1[j]=text[i][3+j];
              ptr1[length]='\0';
              text[i]=realloc(text[i], length+1);
              strcpy(text[i], ptr1);
              continue;
         }
         if(length>0){
              ptr2=(char*)realloc(ptr2, length+1);
              for(int j=0; j<length; j++)
                   ptr2[length-1-j]=text[i][strlen(text[i])-2-j];
              ptr2[length]='\0';
              text[i]=realloc(text[i], length+1);
              strcpy(text[i], ptr2);
              continue;
              }
         if(length==0){
              free(text[i]);
              (*sentence_count)--;
              for(int j=i; j<*sentence_count; j++)
                   text[j]=text[j+1];
              i--;
              continue;
         }
    }
    free(ptr1);
    free(ptr2);
    return;
}
```

Принимает на вход указатель на динамический массив строк и указатель на целое значение количества предложений. Для каждого предложения функция вызывает функцию *isToBe*, и выполняет действия в зависимости от возвращаемого значения этой функции:

• если 0, то удаляет эту строку, освобождает динамическую память и уменьшает значение количества предложений;

• если меньше 0, то строка становится подстрокой №1;

• если больше 0, то строка становится подстрокой №2.

В ходе работы функции изменяется значение количества предложений.

1.6. Функция проверки длины первого слова предложения

```
int isFourWord(char *sentence){
    for(int i=0; i<4; i++)
         if(!isalnum(sentence[i]))
              return 0;
    if(!isalnum(sentence[4]))
         return 1;
    return 0;
}
```

Принимает на вход указатель на строку. Возвращает:

• 1, если первое слово предложения составлено ровно из 4 символов, не являющихся разделителем предложений или слов;

• 0, если иначе.

1.7. Функция удаления предложений при условии длины первого слова предложения

```
void deleteFour(char **text, int *sentence_count){
    for(int i=0; i<*sentence_count; i++)
         if(isFourWord(text[i])){
              free(text[i]);
              (*sentence_count)--;
              for(int j=i; j<*sentence_count; j++)
                   text[j]=text[j+1];
              i--;
         }
    return;
}
```

Принимает на вход указатель на динамический массив строк и указатель на целое значение количества предложений. Для каждого предложения

функция вызывает функцию *isFourWord*, и выполняет действия в зависимости от возвращаемого значения этой функции:

• если 0, то удаляет эту строку, освобождает динамическую память и уменьшает значение количества предложений;

• если 1, то оставляет неизменной.

В ходе работы функции изменяется значение количества предложений.

1.8. Функция разворота символов предложения

```
void backwardText(char **text, int sentence_count){
    int j, k;
    char temp;
    for(int i=0; i<sentence_count; i++){
          j=0;
          k=strlen(text[i])-2;
          while(j<k){
                temp=text[i][j];
                text[i][j]=text[i][k];
                text[i][k]=temp;
                j++;
                k--;
          }
    }
    return;
}
```

Принимает на вход указатель на динамический массив строк и значение количества предложений. Функция разворачивает каждое предложение справа налево (за исключением знаков точки и нулевого символа «\0»).

1.9. Компаратор строк без учёта символов разделителей предложений

```
int mystrcmp(const void* first, const void* second){
    int i1=0;
    int i2=0;
    char *ptr1=*((char**)first);
    char *ptr2 =*((char**)second);
    while(1){
          if((ptr1[i1]=='.')&&(ptr2[i2]=='.'))
                return 0;
          if(ptr1[i1]=='.')
                return -ptr2[i2];
          if((ptr2)[i2]=='.')
                return ptr1[i1];
          if(!isalnum(ptr1[i1]))
                i1++;
          if(!isalnum(ptr2[i2]))
                i2++;
          if(ptr1[i1]==ptr2[i2]){
```

```
                i1++;
                i2++;
            } else return ptr1[i1]-ptr2[i2];
    }
}
```

Принимает на вход 2 указателя на строки. Компаратор лексикографически сравнивает строки, пропуская знаки разделителей предложений (пробел, запятая) до знака разделителя предложений (точка). Возвращает:

- 0, если предложения равны;
- меньше 0, если строка №2 больше строки №1;
- больше 0, если строка №1 больше строки №2.

1.10. Функция сортировки предложений по последнему символу

```
void sortLast(char **text, int sentence_count){
    backwardText(text, sentence_count);
    qsort(text, sentence_count, sizeof(char*), mystrcmp);
    backwardText(text, sentence_count);
    return;
}
```

Принимает на вход указатель на динамический массив строк и значение количества предложений. Функция разворачивает справа налево каждое предложение посредством функции *backwardText*, сортирует предложения по увеличению кода последнего символа, не являющегося разделителем предложений или слов, с помощью функции *sqort* и снова разворачивает каждое предложение справа налево.

1.11. Функция вывода текста

```
void printText(char **text, int sentence_count){
    for(int i=0; i<sentence_count; i++){
        printf("%s\n", text[i]);
        free(text[i]);
    }
    free(text);
    return;
}
```

Принимает на вход указатель на динамический массив строк и значение количества предложений. Функция выводит на экран с новой строки («\0») каждую строку по указателю и освобождает динамическую память.

ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

```
root@DESKTOP-622GR98:~# gcc all.c && ./a.out
Abcde.QwertY,1a23ooOO.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
1

Bbcdf.
QwfrtZ,1b23ppPP.
root@DESKTOP-622GR98:~# gcc all.c && ./a.out
AEIOYUaeioyu,zxcvZXCVe.Looool.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
1

BFJPZVbfjpzv,zxcvZXCVf.
Lppppl.
root@DESKTOP-622GR98:~#
```

Рис. 1. Демонстрация работы функции *changeVowel* подзадачи №1.

```
root@DESKTOP-622GR98:~# gcc all.o && ./a.out
To be programmist or not to fly like a bird.To sleep,sleep,sleep or not to sleep.To be or.or not to be.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
2

fly like a bird
sleep,sleep,sleep
root@DESKTOP-622GR98:~# gcc all.o && ./a.out
To study or not to work.To study or not.I like apples.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
2

study
root@DESKTOP-622GR98:~# gcc all.o && ./a.out
To be or not to.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
2

root@DESKTOP-622GR98:~#
```

Рис. 2. Демонстрация работы функции *findToBe* подзадачи №2.

```
root@DESKTOP-622GR98:~# gcc all.o && ./a.out
Four word.Five word.Nine word.I like apples.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
3

I like apples.
root@DESKTOP-622GR98:~# gcc all.o && ./a.out
One,two,three,four,five.Money money.Agua regia.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
3

One,two,three,four,five.
Money money.
root@DESKTOP-622GR98:~#
```

Рис. 3. Демонстрация работы функции *deleteFour* подзадачи №3.

```
root@DESKTOP-622GR98:~# gcc all.o && ./a.out
AAAAA.AA.A.AAA.AAAA.B.B.B.BBBBB.BB.BBBB.BBB.A.B.CCCC.CC.C.CC.CCC.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
4

A.
AA.
AAA.
AAAA.
AAAAA.
B.
BB.
BBB.
BBBB.
BBBBB.
C.
CC.
CCC.
CCCC.
root@DESKTOP-622GR98:~# gcc all.o && ./a.out
I like apples.I like bananas.I like apples and bananas.apples.bananas.Fruits.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
4

bananas.
I like apples and bananas.
I like bananas.
apples.
I like apples.
Fruits.
root@DESKTOP-622GR98:~#
```

Рис. 4. Демонстрация работы функции *sortLast* подзадачи №4.

```
root@DESKTOP-622GR98:~# gcc all.o && ./a.out
qwerty.asdfg,asdfg.1e3t5y.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
0

root@DESKTOP-622GR98:~# gcc all.o && ./a.out
1,2,3,4,5.qwerty.qwe,qwe,qwe.
Введите одно из чисел ниже, чтобы выбрать действие:

1) Заменить в тексте все гласные буквы на следующую букву в алфавите.
2) Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и для каждого такого предложения вывести подстроку у которой длина больше.
3) Удалить все предложения у которых длина первого слова равняется 4.
4) Отсортировать предложения по увеличению кода последнего символа не являющегося разделителем предложений или слов.
0) Выйти из программы.
5

Неправильный ввод!
root@DESKTOP-622GR98:~#
```

Рис. 5. Демонстрация работы функции для опции выхода из программы и некорректного ввода.

ЗАКЛЮЧЕНИЕ

В ходе работы были изучены основные библиотеки языка С и были получены навыки работы с функциями. В результате работы была разработана программа, которая редактирует текст по заранее заданным правилам и печатает на экран. Для каждой подзадачи были описаны функции для корректной работы программы. Сопоставленные входные и выходные данные программы удовлетворяют условию задания.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Б.В. Керниган, Д.М. Ричи «Язык С», 229 с.

2. База знаний языка С, С++ // CPlusPlus, http://www.cplusplus.com (дата обращения: 05.09.2019 г.)

3. Онлайн-справочник С, С++ // http://www.c-cpp.ru (дата обращения: 04.09.2019 г.)

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: program.c

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

char** readText(int *sentence_count){
    char **text = (char**)calloc(1, sizeof(char*));
    if (text == NULL)
        return NULL;
    int i=0;
    int sentence_max=0;
    char c = getchar();
    do{
        i=0;
        if((text=(char**)realloc(text,
((*sentence_count)+1)*sizeof(char**)))==NULL)
            return NULL;
        do{
              if
((text[*sentence_count]=(char*)realloc(text[*sentence_count],
sizeof(char*)*(i+3)))==NULL)
                    return NULL;
            text[*sentence_count][i]=c;
            i++;
        }while((c=getchar())!='.');
        text[*sentence_count][i]='.';
        text[*sentence_count][i+1]='\0';
        (*sentence_count)++;
         if (i+2>sentence_max)
              sentence_max=i+2;
    }while((c = getchar())!='\n');

    char checker[*sentence_count][sentence_max];
    int j;
    for(i=0; i<*sentence_count; i++){
         strcpy(checker[i], text[i]);
         for(j=0; checker[i][j]; j++)
             checker[i][j]=tolower(checker[i][j]);
    }
    for(i=0; i<*sentence_count; i++)
         for(j=i+1; j<*sentence_count; j++)
              if(!strcmp(checker[i], checker[j])){
                  free(text[j]);
                   (*sentence_count)--;
                   for(int k=j; k<*sentence_count; k++){
                        text[k]=text[k+1];
                        strcpy(checker[k], checker[k+1]);
                   }
                   j--;
              }
    return text;
}
```

```
void printText(char **text, int sentence_count){
    for(int i=0; i<sentence_count; i++){
         printf("%s\n", text[i]);
         free(text[i]);
    }
    free(text);
    return;
}

int isVowel(char ch){
    const char vowels[] = "aeiouyAEIOUY";
    if (strchr(vowels, ch)==NULL)
         return 0;
    return 1;
}

void changeVowel(char **text, int sentence_count){
    for(int i=0; i<sentence_count; i++)
         for(int j=0; j<strlen(text[i]); j++)
              if(isVowel(text[i][j]))
                   text[i][j]++;
    return;
}

int isToBe(char* sentence){
    char* ptr1 = strstr(sentence, "To ");
    char* ptr2 = strstr(sentence, " or not to ");
    int len1, len2;
    if(ptr1==sentence && ptr2!=NULL){
         len1 = ptr2-ptr1-3;
         len2 = sentence+strlen(sentence)-(ptr2+12);
         if(len1>len2)
              return -len1;
         else return len2;
    }
    else return 0;
}

void findToBe(char **text, int *sentence_count){
    int length=0;
    char *ptr1 = (char*)calloc(1, sizeof(char));
    char *ptr2 = (char*)calloc(1, sizeof(char));
    for(int i=0; i<*sentence_count; i++){
         length = isToBe(text[i]);
         if(length<0){
              length = -length;
              ptr1 = (char*)realloc(ptr1, length+1);
              for(int j=0; j<length; j++)
                   ptr1[j]=text[i][3+j];
              ptr1[length]='\0';
              text[i]=realloc(text[i], length+1);
              strcpy(text[i], ptr1);
              continue;
         }
         if(length>0){
              ptr2=(char*)realloc(ptr2, length+1);
```

```
                for(int j=0; j<length; j++)
                        ptr2[length-1-j]=text[i][strlen(text[i])-2-j];
                ptr2[length]='\0';
                text[i]=realloc(text[i], length+1);
                strcpy(text[i], ptr2);
                continue;
                }
        if(length==0){
                free(text[i]);
                (*sentence_count)--;
                for(int j=i; j<*sentence_count; j++)
                        text[j]=text[j+1];
                i--;
                continue;
        }
    }
    free(ptr1);
    free(ptr2);
    return;
}

int isFourWord(char *sentence){
    for(int i=0; i<4; i++)
        if(!isalnum(sentence[i]))
            return 0;
    if(!isalnum(sentence[4]))
        return 1;
    return 0;
}

void deleteFour(char **text, int *sentence_count){
    for(int i=0; i<*sentence_count; i++)
        if(isFourWord(text[i])){
            free(text[i]);
            (*sentence_count)--;
            for(int j=i; j<*sentence_count; j++)
                text[j]=text[j+1];
            i--;
        }
    return;
}

void backwardText(char **text, int sentence_count){
    int j, k;
    char temp;
    for(int i=0; i<sentence_count; i++){
        j=0;
        k=strlen(text[i])-2;
        while(j<k){
            temp=text[i][j];
            text[i][j]=text[i][k];
            text[i][k]=temp;
            j++;
            k--;
        }
    }
    return;
```

```
}

int mystrcmp(const void* first, const void* second){
    int i1=0;
    int i2=0;
    char *ptr1=*((char**)first);
    char *ptr2 =*((char**)second);
    while(1){
         if((ptr1[i1]=='.')&&(ptr2[i2]=='.'))
              return 0;
         if(ptr1[i1]=='.')
              return -ptr2[i2];
         if((ptr2)[i2]=='.')
              return ptr1[i1];
         if(!isalnum(ptr1[i1]))
              i1++;
         if(!isalnum(ptr2[i2]))
              i2++;
         if(ptr1[i1]==ptr2[i2]){
              i1++;
              i2++;
         } else return ptr1[i1]-ptr2[i2];
    }
}

void sortLast(char **text, int sentence_count){
    backwardText(text, sentence_count);
    qsort(text, sentence_count, sizeof(char*), mystrcmp);
    backwardText(text, sentence_count);
    return;
}

int main(){
     int sentence_count=0;
     int command;
     char** text = readText(&sentence_count);
     if(text==NULL){
          printf("Ошибка выделения памяти.\n");
          return 1;
     }
    printf("Введите одно из чисел ниже, чтобы выбрать действие:\n\n1)
Заменить в тексте все гласные буквы на следующую букву в алфавите.\n2)
Найти все предложения вида “To <подстрока1> or not to <подстрока2>” и
для каждого такого предложения вывести подстроку у которой длина
больше.\n3) Удалить все предложения у которых длина первого слова
равняется 4.\n4) Отсортировать предложения по увеличению кода
последнего символа не являющегося разделителем предложений или
слов.\n0) Выйти из программы.\n");
    scanf("%d", &command);
    printf("\n");
    switch(command){
        case 1:
              changeVowel(text, sentence_count);
              printText(text, sentence_count);
              break;
         case 2:
             findToBe(text, &sentence_count);
```

```
                printText(text, sentence_count);
                break;
            case 3:
                deleteFour(text, &sentence_count);
                printText(text, sentence_count);
                break;
            case 4:
                sortLast(text, sentence_count);
                printText(text, sentence_count);
                break;
            case 0:
                break;
            default:
                printf("Неправильный ввод!\n");
                break;
    }
    return 0;
}
```