**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# КУРСОВАЯ РАБОТА

**по дисциплине «Программирование»**

**Тема: Работа со строками в языке Си**

Студент гр. 1304 _____ Кривоченко Д.И.

Преподаватель _____ Чайка К.В.

Санкт-Петербург

2021

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

Студент Кривоченко Д.И.

Группа 1304

Тема работы: Работа со строками си

Исходные данные:
Даны строки, состоящие из слов. Строки состоят из латинских букв и цифр. Ввод строк заканчивается символом переноса строки.

Содержание пояснительной записки:
Содержание.
Введение.
Описание кода программы.
Заключение.
Список используемых источников

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания: 15.07.2021

Дата сдачи реферата: 13.12.2021

Дата защиты реферата: 16.12.2021

Студент гр. 1304 _____ Кривоченко Д.И.

Преподаватель _____ Чайка К.В.

# АННОТАЦИЯ

Работа представляет из себя программу, предполагающую работу с текстом. На вход подаётся текст (текст представляет собой предложения, разделенные точкой. Предложения – набор слов, разделенных пробелом или запятой, слова – набор латинских букв и цифр), длина текста и предложений неизвестна. В программе этот текст сохраняется в динамический массив строк. Программа удаляет повторяющиеся предложения и запрашивает у пользователя одно из четырех действий и выполняет его. Код программы написан на языке Си, запуск выполняется на операционной системе Linux. Для ввода текста и операций над ним использованы написанные функции. Исходный код, результаты тестирования представлены в приложениях.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Цель работы – написание программы для считывания и редактирования строк.

Задание:

Требуется считать текст в динамический массив строк и реализовать следующие функции:

1. Функция, выводящая предложения, в которых все слова состоят только из цифр.
2. Функция, отсортировывающая предложения по уменьшению количества заглавных букв в них.
3. Функция, удаляющая все предложения, в которых какое-то слово встречается 2 и более раза.
4. Функция, преобразовывающая все предложения так, что каждое вхождение цифры заменяется на соответствующую подстроку (0 - “Zero”, 1 – “One” и т.д.)

Также требуется написать функцию, запрашивающую у пользователя одно из доступных действий и выполнять его.

# 1. ОПИСАНИЕ КОДА ПРОГРАММЫ

## 1.1. Функция считывания текста

```
int readArr(char* sentTern, char*** text)
{
    int a = 0;
    char c;
    int flagSent = 1;
    int flagText = 1;
    int sentCount = 0;
    int charCount = 0;
    int string_length = 1000;
    while (1)
    {
        charCount = 0;
        (*text) = realloc((*text), sizeof(char*) * (sentCount+1));
        if ((*text) == NULL){
                puts("Недостаточно памяти!");
                exit(0);
                }
        (*text)[sentCount] = malloc(sizeof(char) * string_length);
        c = getchar();
        if (c == '\n')
        {
            a++;
            if (a==2)
                break;
        }
        else{
                (*text)[sentCount][charCount] = c;
                a=0;
        }
        do
        {
            charCount++;
            c = getchar();
            if ((c == '\n')){
                a++;
                break;
            }
            else{
                a = 0;
            }
            (*text)[sentCount][charCount] = c;
            flagSent = (strchr(sentTern,c) == NULL);
            if (charCount>=string_length - 2)
            {
                string_length = string_length + 500;
                (*text)[sentCount] = (char*)
realloc((*text)[sentCount], string_length);
                if ((*text)[sentCount] == NULL){
                puts("Недостаточно памяти!");
                exit(0);
                }
            }
        }
```

```
        while(flagSent == 1);
        if (a == 2)
            break;
        c=getchar();
        if (c == '\n')
           a++;
        else
           a = 0;
        (*text)[sentCount][charCount+1] = ' ';
        (*text)[sentCount][charCount+2] = '\0';
        sentCount++;
    }
    return sentCount;
}
```

Функция  посимвольно считывает текст, построчно выделяя память под него пока пользователь не прервёт ввод.

### 1.2. Функция удаления одинаковых предложений

```
int deleteSent(int sentCount, char*** text)
{
    int indexCount = 0;
    for(int i =0; i<sentCount; i++)
    {
            int j = i+1;
            while (j<sentCount){
                if (strcasecmp((*text)[i],(*text)[j])==0){
                    free((*text)[j]);
                    sentCount--;
                    for (int f = j; f<=sentCount;f++){
                        (*text)[f] = (*text)[f+1];
                    }
                }
                else{
                    j++;
                }
            }
    }
    return sentCount;
}
```

Функция перебирает все пары предложений, посимвольно сравнивая их с помощью функции strcasecmp. Если предложения одинаковы — сдвигаем на один влево начиная с найденного.

### 1.3. Функция вывода предложений, состоящих только из цифр

```
void printNums(int new_len, char* sentTern,   char*** text)
{
    char k;
    int flag = 0;
    char nums[10] = "0987654321";
    int flag_first_space = 0;
    for (int i =0; i<new_len; i++)
    {
        flag = 1;
        for (int j =0; j<strlen((*text)[i]); j++)
        {
            if ((strchr(sentTern,(*text)[i][j]) == NULL) &&
(strchr(nums,(*text)[i][j]) == NULL))
            {
                flag = 0;
                break;
            }
        }

        if (flag)
        {
            for (int j = 0; j<strlen((*text)[i]);j++){
                if ((flag_first_space==0) && (j==0) && ((*text)[i][j])
== ' '){
                    flag_first_space = 1;
                    continue;
                }
                else{
                    printf("%c", (*text)[i][j]);
                }
            }
        }
    }
    puts("");

}
```

Функция посимвольно проверяет, все ли символы являются цифрами (не считая знаков препинания). Если это так – то она выводит это предложение.

## 1.4. Функция, которая удаляет предложения, в которых встречаются одинаковые слова

```
int delSentwWords(int new_len, char*** text)
{
    int count = 0;
    int moveCount = 0;
    int ma =0;
    int i =0;
    char** res = NULL;
    int flag = 0;
    int indexcount = 0;
    int index[100];
    for (int i=0;i<100;i++){
         index[i] = -1;
    }



    while(i<new_len)
    {
         if (i!=0){
               for (int f = 0; f< count; f++){
                     free(res[f]);
               }
               free(res);
         }
         count = 0;
        res = malloc(100*sizeof(char*));
        if (res == NULL){
         puts("Недостаточно памяти!");
         exit(0);
        }
        char s[strlen((*text)[i])];
        strcpy(s,(*text)[i]);
        char *t = strtok(s, " ,");
        do
        {
            res[count] = malloc(100*1);
            if (res[count] == NULL){
         puts("Недостаточно памяти!");
         exit(0);
```

```
        }
            if (t[strlen(t)-1] == '.')
               t[strlen(t)-1] = '\0';
            strcpy(res[count], t);
            count++;
            t=strtok(NULL, " ,");
        }
        while(t!=NULL);
    //wqe, fds. cxv. qweeqw, asd zxc. wqe, fds. qwe. qwe. kfsdl. zxc, bvc fdg rwe asd zxc.
    //qwe, zxc qwe. vcjxk. qwe, fdb qwe. sdfkj. cvx, vcx. dfsfsdfdsfds. qwe, zxc qwe.
        if (count>1){
            for (int t = 0; t<count;t++){
                if (flag)
                    break;

                for (int j = t+1;j<count;j++){
            //      printf("[%d][%s][%s]\n",i, res[t], res[j]);
                        if ((t!=j) && (strcmp(res[t],res[j])==0)){
                            flag = 1;
                            break;

                            }
                    }
                }
        }
    if (flag){
         index[indexcount++] = i;
         }
    i++;
    flag = 0;
    }
    for (int i = 0;i<indexcount;i++){
         int idx = index[i];
         if (idx != -1){
               moveCount++;
               for (int k = idx; k<new_len-moveCount;k++){
                     strcpy((*text)[k],(*text)[k+1]);
               }
         }
```

```
    }
    int f = new_len - moveCount;
    return f;

}
```

Функция копирует данную строку в новую переменную, разбивает её на слова с помощью функции strtok и записывает их в массив. Далее производится перебор всех пар по массиву, и, если находятся два одинаковых слова – переменная flag принимает значение 1, а индекс этого предложения записывается в массив индексов. Далее производится перебор по массиву индексов и сдвиг предложений.

### 1.5. Функция, которая заменяет цифры в словах на соответствующие подстроки

```
void changeNums(int new_len, char*** text)
{
    char* nums = "1234567890";
    char* dummy_str =(char*) calloc(sizeof(char), 3000);
    if (dummy_str == NULL){
           puts("Недостаточно памяти!");
           exit(0);
        }
    int dummy_count = 0;
    for (int i = 0; i<new_len; i++)
    {
        for (int j =0; j<strlen((*text)[i]); j++)
        {
            if (strchr(nums,(*text)[i][j])!=NULL)
            {
                switchCaseNums(&dummy_str,(*text)[i][j],&dummy_count);

            }
            else
            {
                dummy_str[dummy_count] = (*text)[i][j];
                dummy_count++;
            }

        }

        strcpy((*text)[i],dummy_str);


        free(dummy_str);
```

```
            dummy_str =(char*) calloc(sizeof(char), 3000);
            if (dummy_str == NULL){
               puts("Недостаточно памяти!");
               exit(0);
            }
            dummy_count = 0;
        }
        free(dummy_str);
}
```

Функция создает строку dummy_str, в которую посимвольно перезаписывается исходная строка. Если встречается цифра, вызывается другая функция switchCaseNums, в которой с помощью switch выражения вместо цифры записывается соответствующая подстрока. Далее dummy_str копируется в исходный текст на место соответствующей строки.

## 1.6. Функция, сравнивающая количество заглавных букв в двух строках

```
int cmp(const void* a, const void* b)
{

    char** first = (char**) a;
    char** second = (char**) b;



    char *str1 = *first;
    char *str2 = *second;
    char punctMarks[] = "1234567890.,;:? ";
    int count1 = 0;
    int count2 = 0;
    for (int i =0; i < strlen(str1); i++)
    {
        if ((toupper(str1[i]) == str1[i]) &&
(strchr(punctMarks,str1[i])==NULL))
            count1++;
    }
    for (int i =0; i < strlen(str2); i++)
    {
        if ((toupper(str2[i]) == str2[i]) &&
(strchr(punctMarks,str2[i])==NULL))
            count2++;
    }
    if (count1<count2) return 1;
    if (count1>=count2) return -1;
}
```

Функция написана для сортировки предложений в тексте по количеству заглавных букв, встречающихся в нём. Она сравнивает каждый символ, не учитывая знаки препинания и цифры.

### 1.7. Функция, сортирующая предложения по количеству заглавных букв в них

```
void sortSents(int sentCount, char*** text)
{

    qsort(*text, sentCount, sizeof(char*), cmp);

}
```

Функция сортирует предложения, по количеству заглавных букв в них. Она использует qsort для сортировки, сохраняет “правильный” вид предложений (пробелы после точек, отсутствие лишних пробелов). Функция, подаваемая qsort – cmp.

### 1.8. Функция, выводящая полученный текст

```
void printArr(int sentCount, char*** text)
{
    for (int i =0; i<sentCount; i++)
        printf("%s", (*text)[i]);

}
```

Функция выводит текст с помощью цикла.

### 1.8.2 Функция, освобождающая память, занимаемую текстом

```
void FreeText(int toFree, char*** text){

    for (int i =0;i<toFree;i++)
        free((*text)[i]);
    free(*text);

}
```

Функция освобождает память, занимаемую текстом с помощью цикла.

### 1.8.3 Функция, проверяющая, есть ли данное значение в массиве

```
int inArr(int x, int len, int *arr)
{
    for (int i = 0; i < len; i++)
    {
        if (arr[i] == x)
        {
            return 1;
            break;
        }
    }
    return 0;
}
```

## 1.9. Главная функция

```
int main()
{

    char** text = malloc(sizeof(char*));
    char* sentTern = ".";
    char* sentTernExtended = "., ";
    puts("Если вы желаете прекратить ввод, введите пустую строку.");
    puts("-------------------------------------------------------");
    int sentCount = readArr(sentTern, &text);
    int new_len = deleteSent(sentCount, &text);
    int userChoice;

    puts("-------------------------------------------------------------
----------");
    puts("Выберите, что вы хотите сделать с текстом:");
    puts("1) Напечатать все предложения, состоящие только из цифр");
    puts("2) Отсортировать предложения по количеству заглавных букв
в них");
    puts("3) Удалить все предложения, в которых есть 2 и более
одинаковых слова");
    puts("4) Заменить все цифры на их текстовые написания");
    puts("Если вы хотите выйти из программы, введите любой другой
символ");
    printf("Пожалуйста, введите ваш выбор ");
    scanf("%d", &userChoice);
    puts("-------------------------------------------------------------
-----------");



    switch(userChoice)
    {
    case 1:
        printNums(new_len, sentTernExtended, &text);
        break;
    case 2:
        sortSents(new_len, &text);
        printArr(new_len, &text);
        break;
    case 3:
        new_len = delSentwWords(new_len, &text); //returns new len
according to an amount of deleted sents
        printArr(new_len, &text);
        break;
    case 4:
        changeNums(new_len, &text);
        printArr(new_len, &text);
        break;
    default:
        break;
    }

    FreeThem(new_len, &text);

    return 0;
}
```

Функция выводит графический интерфейс, считывает и обрабатывает ввод пользователя и вызывает остальные функции в зависимости от него.

# ЗАКЛЮЧЕНИЕ

Для успешного написания программы для работы с текстом, соответствующей заданию курсовой работы, были выполнены следующие задачи:

1. Изучен теоретический материал по теме курсовой работы.
2. Разработан и реализован программный код.
3. Проведено тестирование программы.

Исходный код программы представлен в приложении А, результаты тестирования - в приложении Б.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. The C Programming Language / Brian W. Kernigan, Dennis M. Ritchie Second edition, 1988. 288 c.

2. The C++ Resources network [Электронный ресурс]
URL: http://cplusplus.com

# ПРИЛОЖЕНИЕ А
# ИСХОДНЫЙ КОД ПРОГРАММЫ

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



int inArr(int x, int len, int* arr);
int readArr(char* sentTern, char*** text);
int deleteSent(int sentCount, char*** text);
int cmp(const void* a, const void* b);
void printArr(int sentCount, char*** text);

void printNums(int new_len, char* sentTern,   char*** text);
int delSentwWords(int new_len, char*** text);

void switchCaseNums(char** dummy_str, char text_i_j, int*
dummy_count);
void changeNums(int new_len, char*** text);
void sortSents(int sentCount, char*** text);


void FreeThem(int toFree, char*** text);

int main()
{

    char** text = malloc(sizeof(char*));
    char* sentTern = ".";
    char* sentTernExtended = "., ";
    puts("Если вы желаете прекратить ввод, введите пустую
строку.");
    puts("-----------------------------------------------------
");
    int sentCount = readArr(sentTern, &text);
    int new_len = deleteSent(sentCount, &text);
    int userChoice;

    puts("-----------------------------------------------------
------------");
```

```
    puts("Выберите, что вы хотите сделать с текстом:");
    puts("1) Напечатать все предложения, состоящие только из
цифр");
    puts("2) Отсортировать предложения по количеству заглавных
букв в них");
    puts("3) Удалить все предложения, в которых есть 2 и более
одинаковых слова");
    puts("4) Заменить все цифры на их текстовые написания");
    puts("Если вы хотите выйти из программы, введите любой другой
символ");
    printf("Пожалуйста, введите ваш выбор ");
    scanf("%d", &userChoice);
    puts("-------------------------------------------------------
-------------");



    switch(userChoice)
    {
    case 1:
        printNums(new_len, sentTernExtended, &text);
        break;
    case 2:
        sortSents(new_len, &text);
        printArr(new_len, &text);
        break;
    case 3:
        new_len = delSentwWords(new_len, &text); //returns new len
according to an amount of deleted sents
        printArr(new_len, &text);
        break;
    case 4:
        changeNums(new_len, &text);
        printArr(new_len, &text);
        break;
    default:
        break;
    }

    FreeThem(new_len, &text);

    return 0;
}
```

```
int inArr(int x, int len, int *arr)
{
    for (int i = 0; i < len; i++)
    {
        if (arr[i] == x)
        {
            return 1;
            break;
        }
    }
    return 0;
}
// I hasbh, sdvji. Q fsdjio vxc. I sadmn. HerE It. dfskdsfdfs.
mbbvc, bvc.
// qwe, asd, zcx, hgfk. erioptrepoi, cvx. sdf. qwe, asd, zcx,
hgfk. bvcm, sfd. cvx. EOL.
// qwe, asd zxc. dgfglkfjd, qwe, asd, zxc. gfdkj, eol eol. cxvnm.
qwe, asd zxc. 24389Five.

int readArr(char* sentTern, char*** text)
{
    int a = 0;
    char c;
    int flagSent = 1;
    int flagText = 1;
    int sentCount = 0;
    int charCount = 0;
    int string_length = 1000;
    while (1)
    {
        charCount = 0;
        (*text) = realloc((*text), sizeof(char*) * (sentCount+1));
        (*text)[sentCount] = malloc(sizeof(char) * string_length);
```

```
        c = getchar();
        if (c == '\n')
        {
            a++;
            if (a==2)
               break;
        }
        else{
                (*text)[sentCount][charCount] = c;
                a=0;
        }
        do
        {
            charCount++;
            c = getchar();
            if ((c == '\n')){
               a++;
                break;
            }
            else{
                a = 0;
            }
            (*text)[sentCount][charCount] = c;
            flagSent = (strchr(sentTern,c) == NULL);
            if (charCount>=string_length - 2)
            {
                string_length = string_length + 500;
                (*text)[sentCount] = (char*)
realloc((*text)[sentCount], string_length);
            }
        }
        while(flagSent == 1);
        if (a == 2)
            break;
        c=getchar();
        if (c == '\n')
          a++;
        else
          a = 0;
        (*text)[sentCount][charCount+1] = ' ';
        (*text)[sentCount][charCount+2] = '\0';
        sentCount++;
    }
    return sentCount;
}
```

```
int deleteSent(int sentCount, char*** text)
{
    for(int i =0; i<sentCount; i++)
    {

            int j = i+1;

            while (j<sentCount){
                if (strcasecmp((*text)[i],(*text)[j])==0){
               //     printf("[%s][%s][%d][%d]\n", (*text)[i],
(*text)[j],i,j);
                    free((*text)[j]);
                    sentCount--;
                    for (int f = j; f<=sentCount;f++){
                   //    printf("[%s][%d]\n",(*text)[f+1],f+1);
                        (*text)[f] = (*text)[f+1];
                    }
                }
                else{
                    j++;
                }
            }
    }
    return sentCount;
}
```

```
void printNums(int new_len, char* sentTern,   char*** text)
{
    char k;
    int flag = 0;
    char nums[10] = "0987654321";
    int flag_first_space = 0;
    for (int i =0; i<new_len; i++)
    {
        flag = 1;
        for (int j =0; j<strlen((*text)[i]); j++)
        {
            if ((strchr(sentTern,(*text)[i][j]) == NULL) &&
(strchr(nums,(*text)[i][j]) == NULL))
```

```
            {
                flag = 0;
                break;
            }
        }

        if (flag)
        {
            for (int j = 0; j<strlen((*text)[i]);j++){
                if ((flag_first_space==0) && (j==0) &&
((*text)[i][j]) == ' '){
                    flag_first_space = 1;
                    continue;
                }
                else{
                    printf("%c", (*text)[i][j]);
                }
            }
        }
    }
    puts("");

}
```

```
int delSentwWords(int new_len, char*** text)
{
    int count = 0;
    int moveCount = 0;
    int ma =0;
    int i =0;
    char** res = NULL;
    int flag = 0;
    int indexcount = 0;
    int index[100];
    for (int i=0;i<100;i++){
     index[i] = -1;
    }



    while(i<new_len)
```

```
    {
     if (i!=0){
          for (int f = 0; f< count; f++){
               free(res[f]);
          }
          free(res);
     }
     count = 0;
        res = malloc(100*sizeof(char*));
        char s[strlen((*text)[i])];
        strcpy(s,(*text)[i]);
        char *t = strtok(s, " ,");
        do
        {
            res[count] = malloc(100*1);
            if (t[strlen(t)-1] == '.')
               t[strlen(t)-1] = '\0';
            strcpy(res[count], t);
            count++;
            t=strtok(NULL, " ,");
        }
        while(t!=NULL);
    //wqe, fds. cxv. qweeqw, asd zxc. wqe, fds. qwe. qwe. kfsdl.
zxc, bvc fdg rwe asd zxc.
    //qwe, zxc qwe. vcjxk. qwe, fdb qwe. sdfkj. cvx, vcx.
dfsfsdfdsfds. qwe, zxc qwe.
        if (count>1){
            for (int t = 0; t<count;t++){
                if (flag)
                    break;

                for (int j = t+1;j<count;j++){
            //      printf("[%d][%s][%s]\n",i, res[t], res[j]);
                        if ((t!=j) && (strcmp(res[t],res[j])==0)){
                            flag = 1;
                            break;

                            }
                    }
                }
        }
    if (flag){
     index[indexcount++] = i;
     }
    i++;
    flag = 0;
```

```
    }
    for (int i = 0;i<indexcount;i++){
     int idx = index[i];
     if (idx != -1){
          moveCount++;
          for (int k = idx; k<new_len-moveCount;k++){
               strcpy((*text)[k],(*text)[k+1]);
          }
     }
    }
    int f = new_len - moveCount;
    return f;

}


void switchCaseNums(char** dummy_str, char text_i_j, int*
dummy_count)
{
    switch(text_i_j)
    {
    case '0':
        (*dummy_str)[(*dummy_count)++] = 'Z';
        (*dummy_str)[(*dummy_count)++] = 'e';
        (*dummy_str)[(*dummy_count)++] = 'r';
        (*dummy_str)[(*dummy_count)++] = 'o';
        break;
    case '1':
        (*dummy_str)[(*dummy_count)++] = 'O';
        (*dummy_str)[(*dummy_count)++] = 'n';
        (*dummy_str)[(*dummy_count)++] = 'e';
        break;
    case '2':
        (*dummy_str)[(*dummy_count)++] = 'T';
        (*dummy_str)[(*dummy_count)++] = 'w';
        (*dummy_str)[(*dummy_count)++] = 'o';
        break;
    case '3':
        (*dummy_str)[(*dummy_count)++] = 'T';
        (*dummy_str)[(*dummy_count)++] = 'h';
        (*dummy_str)[(*dummy_count)++] = 'r';
        (*dummy_str)[(*dummy_count)++] = 'e';
        (*dummy_str)[(*dummy_count)++] = 'e';
        break;
    case '4':
        (*dummy_str)[(*dummy_count)++] = 'F';
```

```
            (*dummy_str)[(*dummy_count)++] = 'o';
            (*dummy_str)[(*dummy_count)++] = 'u';
            (*dummy_str)[(*dummy_count)++] = 'r';
            break;
        case '5':
            (*dummy_str)[(*dummy_count)++] = 'F';
            (*dummy_str)[(*dummy_count)++] = 'i';
            (*dummy_str)[(*dummy_count)++] = 'v';
            (*dummy_str)[(*dummy_count)++] = 'e';

            break;
        case '6':
            (*dummy_str)[(*dummy_count)++] = 'S';
            (*dummy_str)[(*dummy_count)++] = 'i';
            (*dummy_str)[(*dummy_count)++] = 'x';
            break;
        case '7':
            (*dummy_str)[(*dummy_count)++] = 'S';
            (*dummy_str)[(*dummy_count)++] = 'e';
            (*dummy_str)[(*dummy_count)++] = 'v';
            (*dummy_str)[(*dummy_count)++] = 'e';
            (*dummy_str)[(*dummy_count)++] = 'n';
            break;
        case '8':
            (*dummy_str)[(*dummy_count)++] = 'E';
            (*dummy_str)[(*dummy_count)++] = 'i';
            (*dummy_str)[(*dummy_count)++] = 'g';
            (*dummy_str)[(*dummy_count)++] = 'h';
            (*dummy_str)[(*dummy_count)++] = 't';
            break;
        case '9':
            (*dummy_str)[(*dummy_count)++] = 'N';
            (*dummy_str)[(*dummy_count)++] = 'i';
            (*dummy_str)[(*dummy_count)++] = 'n';
            (*dummy_str)[(*dummy_count)++] = 'e';
            break;
        }
    }

    void changeNums(int new_len, char*** text)
    {
        char* nums = "1234567890";
        char* dummy_str =(char*) calloc(sizeof(char), 10000);
        int dummy_count = 0;
        for (int i = 0; i<new_len; i++)
        {
```

```
        for (int j =0; j<strlen((*text)[i]); j++)
        {
            if (strchr(nums,(*text)[i][j])!=NULL)
            {

switchCaseNums(&dummy_str,(*text)[i][j],&dummy_count);

            }
            else
            {
                dummy_str[dummy_count] = (*text)[i][j];
                dummy_count++;
            }

        }

        strcpy((*text)[i],dummy_str);


        free(dummy_str);
        dummy_str =(char*) calloc(sizeof(char), 10000);
        dummy_count = 0;
    }
    free(dummy_str);
}


int cmp(const void* a, const void* b)
{

    char** first = (char**) a;
    char** second = (char**) b;



    char *str1 = *first;
    char *str2 = *second;
    char punctMarks[] = "1234567890.,;:? ";
    int count1 = 0;
    int count2 = 0;
    for (int i =0; i < strlen(str1); i++)
    {
        if ((toupper(str1[i]) == str1[i]) &&
(strchr(punctMarks,str1[i])==NULL))
            count1++;
    }
```

```
    for (int i =0; i < strlen(str2); i++)
    {
        if ((toupper(str2[i]) == str2[i]) &&
(strchr(punctMarks,str2[i])==NULL))
            count2++;
    }
    if (count1<count2) return 1;
    if (count1>=count2) return -1;
}

//I hasbh, sdvji. Q fsdjio vxc. I sadmn. HerE It. dfskdsfdfs.
mbbvc, bvc.

void sortSents(int sentCount, char*** text)
{

    qsort(*text, sentCount, sizeof(char*), cmp);

}

void printArr(int sentCount, char*** text)
{
    for (int i =0; i<sentCount; i++)
        printf("%s", (*text)[i]);
    puts("");
}

void FreeThem(int toFree, char*** text){
    for (int i =0;i<toFree;i++)
        free((*text)[i]);
    free(*text);
}
```

# ПРИЛОЖЕНИЕ Б

# РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ

```
dmitry@HP-Dmitry:~/Рабочий стол$ gcc main.c && ./a.out
Если вы желаете прекратить ввод, введите пустую строку.
------------------------------------------------------
qwe, zxc qwe. vcjxk. qwe, fdb qwe. 2433424, 123. 123WEQWEQ213. sdfkj. cvx, vcx. dfsfsdfdsfds. qwe, zxc qwe.

------------------------------------------------------------------
Выберите, что вы хотите сделать с текстом:
1) Напечатать все предложения, состоящие только из цифр
2) Отсортировать предложения по количеству заглавных букв в них
3) Удалить все предложения, в которых есть 2 и более одинаковых слова
4) Заменить все цифры на их текстовые написания
Если вы хотите выйти из программы, введите любой другой символ
Пожалуйста, введите ваш выбор 3
------------------------------------------------------------------
vcjxk. qwe, fdb qwe. 123WEQWEQ213. sdfkj. cvx, vcx. dfsfsdfdsfds.
dmitry@HP-Dmitry:~/Рабочий стол$ gcc main.c && ./a.out
Если вы желаете прекратить ввод, введите пустую строку.
------------------------------------------------------
qwe, zxc qwe. vcjxk. qwe, fdb qwe. 2433424, 123. 123WEQWEQ213. sdfkj. cvx, vcx. dfsfsdfdsfds. qwe, zxc qwe.

------------------------------------------------------------------
Выберите, что вы хотите сделать с текстом:
1) Напечатать все предложения, состоящие только из цифр
2) Отсортировать предложения по количеству заглавных букв в них
3) Удалить все предложения, в которых есть 2 и более одинаковых слова
4) Заменить все цифры на их текстовые написания
Если вы хотите выйти из программы, введите любой другой символ
Пожалуйста, введите ваш выбор 4
------------------------------------------------------------------
qwe, zxc qwe. vcjxk. qwe, fdb qwe. TwoFourThreeThreeFourTwoFour, OneTwoThree. OneTwoThreeWEQWEQTwoOneThree. sdfkj. cvx, vcx. dfsfsdfdsfds.
dmitry@HP-Dmitry:~/Рабочий стол$ gcc main.c && ./a.out
Если вы желаете прекратить ввод, введите пустую строку.
------------------------------------------------------
qwe, zxc qwe. vcjxk. qwe, fdb qwe. 2433424, 123. 123WEQWEQ213. sdfkj. cvx, vcx. dfsfsdfdsfds. qwe, zxc qwe.

------------------------------------------------------------------
Выберите, что вы хотите сделать с текстом:
1) Напечатать все предложения, состоящие только из цифр
2) Отсортировать предложения по количеству заглавных букв в них
3) Удалить все предложения, в которых есть 2 и более одинаковых слова
4) Заменить все цифры на их текстовые написания
Если вы хотите выйти из программы, введите любой другой символ
Пожалуйста, введите ваш выбор 1
------------------------------------------------------------------
2433424, 123.
```

```
dmitry@HP-Dmitry:~/Рабочий стол$ gcc main.c && ./a.out
Если вы желаете прекратить ввод, введите пустую строку.
------------------------------------------------------
I hasbh, sdvji. Q fsdjio vxc. I sadmn. HerE It. dfskdsfdfs. mbbvc, bvc.

------------------------------------------------------------------------
Выберите, что вы хотите сделать с текстом:
1) Напечатать все предложения, состоящие только из цифр
2) Отсортировать предложения по количеству заглавных букв в них
3) Удалить все предложения, в которых есть 2 и более одинаковых слова
4) Заменить все цифры на их текстовые написания
Если вы хотите выйти из программы, введите любой другой символ
Пожалуйста, введите ваш выбор 2
------------------------------------------------------------------------
HerE It. I hasbh, sdvji. Q fsdjio vxc. I sadmn. dfskdsfdfs. mbbvc, bvc.
```