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

# КУРСОВАЯ РАБОТА

**по дисциплине «Программирование»**

**Тема: Обработка строк на языке С**

Студент гр. 0382 _____ Литягин С. М.

Жангиров Т. Р.

Преподаватель _____ Чайка К. В.

Санкт-Петербург

2020

**ЗАДАНИЕ**

**НА КУРСОВУЮ РАБОТУ**

Студент Литягин С. М.

Группа 0382

Тема работы: Обработка строк на языке С.

Вариант 21

Исходные данные:

Программе на вход подается текст (текст представляет собой предложения, разделенные точкой). Предложения - набор слов, разделенные пробелом или запятой, слова - набор латинских букв и цифр, а также символ ‘-’. Длина текста и каждого предложения заранее не известна.

Программа должна сохранить этот текст в динамический массив строк и оперировать далее только с ним.

Программа должна найти и удалить все повторно встречающиеся предложения (сравнивать их следует посимвольно, но без учета регистра).

Далее, программа должна запрашивать у пользователя одно из следующих доступных действий (программа должна печатать для этого подсказку. Также следует предусмотреть возможность выхода из программы):

Отсортировать все предложение по уменьшению суммарного времени указанного в строке. Время в строке указывается в виде слова “HH-MM-SS”. Например, для строки “abs 12-03-13 p 01-10-23 02-32-24” суммарное время составляет “15-46-00”. Если в предложении не указано время, то суммарное время для него составляет “00-00-00”.

Вывести все предложения, в которых количество слов больше 2 и меньше 7.

Во всем тексте заменить согласные строчные буквы на заглавные.

Удалить все предложения, в которых количество слов больше среднего количества слов предложениях.

Все сортировки должны осуществляться с использованием функции стандартной библиотеки. Использование собственных функций, при наличии аналога среди функций стандартной библиотеки, запрещается.

Все подзадачи, ввод/вывод должны быть реализованы в виде отдельной функции.

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 02.11.2020

Дата сдачи реферата: 24.12.2020

Дата защиты реферата: 26.12.2020

Студент _____ Литягин С. М.

Преподаватель _____ Жангиров Т. Р.
Чайка К. В.

# АННОТАЦИЯ

В ходе выполнения курсовой работы была создана программа для обработки введенного текста согласно выбранному пользователем способу. Для хранения текста память выделяется динамически.

Для того чтобы пользователь знал возможные способы обработки текста, выводится сообщение, содержащее информацию о них. Если пользователь выберет несуществующую команду, то выведется ошибка, и ему будет предложено выбрать другую команду.

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

В данной курсовой работе было разработано консольное приложение, созданное для изменения текста, введенного пользователем, в соответствии с выбранным способом обработки. Память для хранения текста выделяется динамически. Чтобы это реализовать - используется стандартный заголовочный файл <stdlib.h>.

Разработка программы велась на операционной системе Ubuntu 20.04 при помощи текстового редактора Atom, а также среды разработки CLion.

# 1. ПОСТАВЛЕННЫЕ ЦЕЛИ И ЗАДАЧИ

Цель работы – создание консольного приложения для обработки текста по запросу пользователя.

Для выполнения работы необходимо выполнить следующие задачи:

1. Ввод, последующее хранение, вывод текста
2. Реализовать сортировку текста согласно суммарному времени, указанному в его предложениях
3. Реализовать сортировку текста согласно определенному количеству слов ($2 < x < 7$)
4. Реализовать замену строчных согласных букв текста на заглавные
5. Реализовать сортировку текста согласно среднему количеству слов в нем.

## 2. ВЫПОЛНЕНИЕ РАБОТЫ

### 2.1. Считывание текста

В начале работы программы пользователю сообщают о возможностях обработки текста, а затем предлагают ввести его. Для этого реализованы три функции:

Функция *char* sentence()*. Эта функция считывает ровно одно предложение. Сначала происходит выделение памяти методом *calloc* для предложения *sent* с первоначальным размером *size_s* = 180. В случае если количество символов, введенных в предложение с помощью функции *getchar()*, становится равным *size_s*, то происходит увеличение выделяемой памяти методом *realloc*. Считывание символов прекращается, когда попадается точка. Если был введен символ окончания ввода ‘\0’, то функция экстренно завершает ввод символов в sent. Функция возвращает считанное предложение.

Функция *char* delete_space(char* sent)*. Эта функция позволяет удалить все пробелы, табуляции и переносы строк в начале предложения *sent*. Функция возвращает измененное предложение *sent*.

Функция *char** input(char** text, int* sent_before)*. Эта функция заполняет *text* с выделенной динамически памятью предложениями, полученными функцией *sentence()*. Однако, до заполнения у предложений удаляются все пробелы, табуляции и переводы строк в начале предложения с помощью функции *delete_space(sent)*. Если последний элемент введенного предложения является символом окончания ввода (‘\0’), то заполнение *text* прекращается. Функция возвращает текст, хранящийся в *text*.

### 2.2. Удаление повторяющихся предложений

Одно из условий задания – удаление повторяющихся предложений в тексте. Для этого написана функция *char** delete_repeat(char** text, &sent_before)*. В этой функции происходит сравнение предложений при помощи функции *strcasecmp()*, которая выполняет побайтовое сравнение строк,

игнорируя регистр символов. Это функция стандартной библиотеки <*string.h*>. Если какое-то из предложений является повторением предложения, которые располагается до него, то тогда его удаляют. Функция возвращает измененный *text*.

### 2.3. Выбор обработки текста

Для дальнейшей работы пользователя просят ввести номер обработки текста из предложенных. Для осуществления дальнейших команд пользователя была создана функция *void menu(char** text, int* sent_before, int n)*. Какая будет выполнена операция определяется конструкцией *switch* в соответствии со значением *n* (1-5 выполнит какие-то действия, другие значения приведут к выводу ошибки пользователю с просьбой ввести корректное значение). Также функция позволяет пользователю и в дальнейшем обрабатывать текст (т.е. дает возможность последовательно применять различные обработки к введенному тексту).

### 2.4. Сортировка текста согласно суммарному времени

Первый вариант обработки введенного текста – сортировка согласно суммарному времени, указанному в предложениях. Для этого была написана функция *char** sortirovka_time(char** text, int* sent_before)*. Эта функция находит в предложении все строки формата “HH-MM-SS”, где H – часы, M – минуты, S – секунды. Она переделывает HH, MM, SS в целочисленный тип и вносит их в специально созданный массив times[i] , где i – индекс предложения, из которого брались значения времени. Затем происходит уменьшение секунд, путем переноса целой части от деления на 60 в минуты. А потом происходит уменьшение минут тем же путем. В конечном итоге формируется новый массив total_time, содержащий целое число, соответствующее записи времени в предложении (т.е. если было написано “43-54-63”, то будет получено значение 435503). Если в предложении не было строк нужного формата, то будет получено значение 0.

В конце функция производит  сортировку по убыванию массива total_time, а вместе с ним и переставляет соответствующие предложения в text.  Функция возвращает отсортированный по уменьшению суммарного времени текст.

### 2.5. Сортировка текста согласно определенному количеству слов

Второй вариант сортировки введенного текста – сортировка текста по количеству слов в предложениях. Если слов больше 2 и меньше 7, то предложение следует оставить, иначе - удалить. Для этих целей написана функция *char** words(char** text, int* sent_before)*. Эта функция рассчитывает количество слов в предложении по пробелам. Если их количество удовлетворяет условию, то предложение остается, иначе – удаляется. Функция возвращает отсортированный *text*.

### 2.6. Замена строчных согласных букв текста на заглавные

Третий вариант изменения введенного текста – все строчные согласные буквы заменяются на заглавные. Для этих целей написана функция  *char** consonant_up(char** text, int* sent_before)*. В ней объявлен массив, содержащий все согласные буквы. Если символ предложения есть в этом массиве, то он является согласным, а тогда он меняется на заглавный с помощью функции *toupper()*. Эта функция стандартной библиотеки *<ctype.h>*. Функция возвращает измененный *text*.

### 2.7. Сортировка текста согласно среднему количеству слов

Четвертый вариант сортировки введенного текста – удалить предложения, в которых слов больше, чем среднее количество слов в предложениях в тексте. Для этого написаны две функции:

Функция *float average_words(char** text, int* sent_before)*. Эта функция нужна для подсчета среднего количества слов в предложениях текста. Она считает количество всех слов в тексте, а затем делит это число на количество предложений, тем самым мы получаем искомое значение. Функция возвращает среднее количество слов в предложениях текста.

Функция *char** delete_less_average_words(char** text, int* sent_before)*. Эта функция находит количество слов в одном предложении, сравнивает это число с полученным значением из функции *average_words()*. И если количество слов в предложении больше, чем среднее количество слов в предложениях , то предложение удаляется. Функция возвращает отсортированный *text*.

### 2.8. Вывод текста

Вывод текста происходит при помощи функции *void output(char** text, &sent_before)*. Здесь выводится каждое предложение через функцию *printf()* стандартной библиотеки <*stdio.h*>. Память высвобождается в *main*, после того, как пользователь завершает работу.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы было создано приложение для обработки текста согласно запросам пользователя. Все задачи были успешно реализованы: программа сортирует тест по убыванию суммарного времени; программа оставляет лишь предложения, в которых слов больше 2 и меньше 7; программа заменяет все строчные согласные на заглавные согласные; программа сортирует текст согласно среднему количеству слов в нем. Таким образом, цель была достигнута.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1.Керниган Б и Ритчи Д. Язык программирования Си. М.: Вильямс, 1978 288с.

## ПРИЛОЖЕНИЕ А

## ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Запуск исполняемого файла:

```
cruelcookie@cruelcookie-VB:/media/sf_ForStudy$ gcc -Wall -o main main.c
cruelcookie@cruelcookie-VB:/media/sf_ForStudy$ ./main
        Здравствуйте, дорогой пользователь!
Вам доступны следующие действия по обработке текста:
  1.Отсортировать все предложение по уменьшению суммарного времени указанного в
строке.
  2.Вывести все предложения в которых количество слов больше 2 и меньше 7.
  3.Во всем тексте заменить согласные строчные буквы на заглавные.
  4.Удалить все предложения в которых количество слов больше среднего количества
 слов предложениях.
  5.Прекращение работы программы.
Введите ваш текст:
```

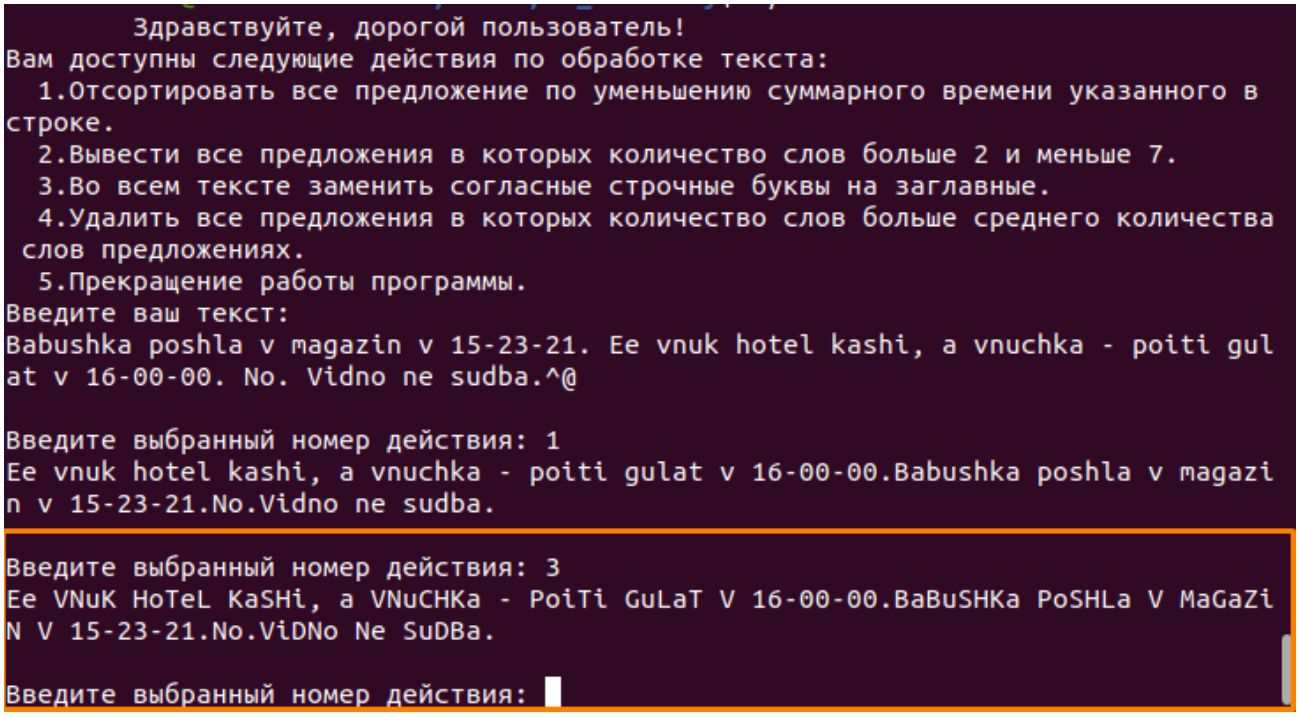
Ввод текста:

```
cruelcookie@cruelcookie-VB:/media/sf_ForStudy$ gcc -Wall -o main main.c
cruelcookie@cruelcookie-VB:/media/sf_ForStudy$ ./main
        Здравствуйте, дорогой пользователь!
Вам доступны следующие действия по обработке текста:
  1.Отсортировать все предложение по уменьшению суммарного времени указанного в
строке.
  2.Вывести все предложения в которых количество слов больше 2 и меньше 7.
  3.Во всем тексте заменить согласные строчные буквы на заглавные.
  4.Удалить все предложения в которых количество слов больше среднего количества
 слов предложениях.
  5.Прекращение работы программы.
Введите ваш текст:
Babushka poshla v magazin v 15-23-21. Ee vnuk hotel kashi, a vnuchka - poiti gul
at v 16-00-00. No. Vidno ne sudba.^@

Введите выбранный номер действия:
```

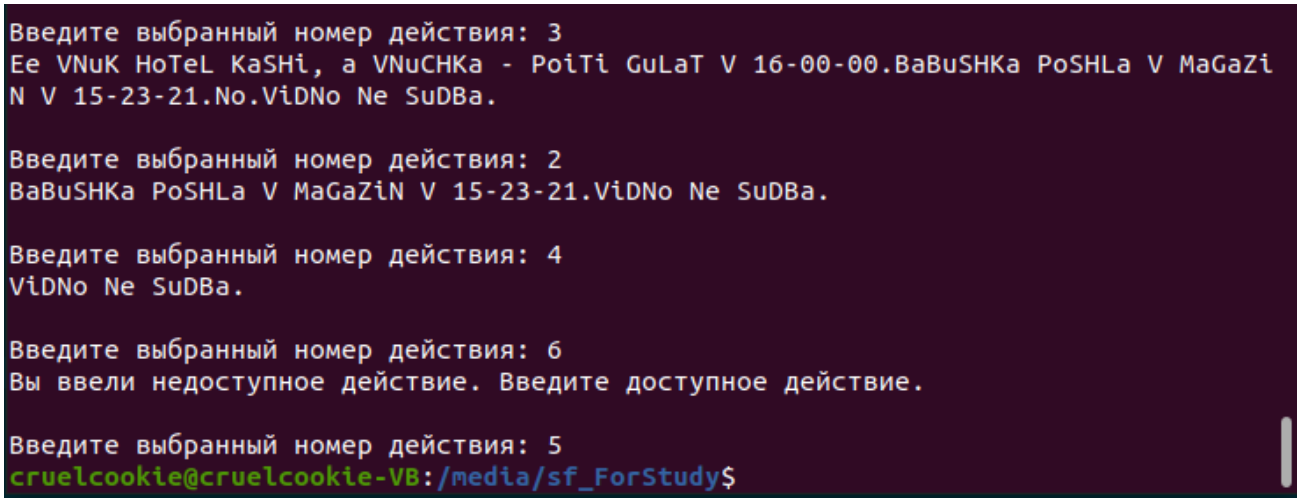
Вывод после обработки первой сортировкой:

```
        Здравствуйте, дорогой пользователь!
Вам доступны следующие действия по обработке текста:
  1.Отсортировать все предложение по уменьшению суммарного времени указанного в
строке.
  2.Вывести все предложения в которых количество слов больше 2 и меньше 7.
  3.Во всем тексте заменить согласные строчные буквы на заглавные.
  4.Удалить все предложения в которых количество слов больше среднего количества
 слов предложениях.
  5.Прекращение работы программы.
Введите ваш текст:
Babushka poshla v magazin v 15-23-21. Ee vnuk hotel kashi, a vnuchka - poiti gul
at v 16-00-00. No. Vidno ne sudba.^@

Введите выбранный номер действия: 1
Ee vnuk hotel kashi, a vnuchka - poiti gulat v 16-00-00.Babushka poshla v magazi
n v 15-23-21.No.Vidno ne sudba.

Введите выбранный номер действия:
```

Вывод после обработки третьей сортировкой:




Выводы после обработки второй и четвертой сортировками, пример ввода недействительной команды, а также команды завершения работы:



# ПРИЛОЖЕНИЕ А
# ПРОГРАММНЫЙ КОД

Файл: main.c

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

//-------<Ввод предложений>-----------------------------------------
-------

char* sentence(){
    int size_s = 180;
    int index = 0;
    char* sent;
    char c = 'f';
    sent = calloc(size_s, sizeof(char));
    while(c != '.'){
        c = getchar();
        sent[index] = c;
        index++;
        if(c == '\0'){
            break;
        }
        if(index == size_s){
            size_s += 50;
            sent = realloc(sent, size_s);
        }
    }
    return sent;
}

//--------<Удаление пробелов, табуляций, переносов строк в начале
предложения>---

char* delete_space(char* sent){
    while(sent[0] == ' ' || sent[0] == '\t' || sent[0] == '\n'){
```

```
        for(int i = 0; i < strlen(sent)-1; i++){
            sent[i] = sent[i+1];
            if (i == strlen(sent) - 2){
                sent[strlen(sent)-1] = '\0';
            }
        }
    }
    return sent;
}

//-------<Удаление повторяющихся предложений>-----------------------------
-------

char** delete_repeat(char** text, int* sent_before){
    for(int i = 0; i < *sent_before; i++){
      for(int j = i+1; j < *sent_before; j++){
        if(!strcasecmp(text[i], text[j])){
          for(int k = j; k < *sent_before; k++){
              text[k] = text[k+1];
          }
          j--;
          *sent_before = *sent_before -1;
        }
      }
    }
    return text;
}

//-------<Отбор предложений, в которых больше 2 и меньше 7 слов>---------
-------

char** words(char** text, int* sent_before){
    int spaces = 0;
    for(int i = 0; i < *sent_before-1; i++){
      for(int j = 0; j < strlen(text[i]); j++){
        if(text[i][j] == ' '){
          spaces ++;
        }
```

```
      }
      if (spaces < 2 || spaces >=6){
        for(int p = i; p < *sent_before; p++){
          text[p] = text[p+1];
        }
        i--;
        *sent_before = *sent_before - 1;
      }
      spaces = 0;
    }
    return text;
}

//-------<Ввод текста, состоящего из предложений>-------------------------
-------

char** input(int* sent_before){
    char** text;
    int size_t = 20;
    char* sent;
    text = malloc(size_t*sizeof(char*));
    while(1){
        sent = sentence();
        *sent_before += 1;
        text[*sent_before-1] = delete_space(sent);
        if(sent[strlen(sent)-1]=='\0'){
            break;
        }
        if(*sent_before == size_t-1){
            size_t += 20;
            text = realloc(text, size_t*sizeof(char*));
        }
    }
    return text;
}

//-------<Удаление предложений с кол-вом слов, больше среднего>----------
-------
```

```
float average_words(char** text, int* sent_before){
  int spaces = 0;
  float words = 0;
  for(int i = 0; i < *sent_before-1; i++){
    for(int j = 0; j < strlen(text[i]); j++){
      if(text[i][j] == ' '){
        spaces ++;
      }
    }
    words = words + spaces + 1;
    spaces = 0;
  }
  words = words/ (*sent_before-1);
  return words;
}

char** delete_less_average_words(char** text, int* sent_before){
    float aver_w = average_words(text, sent_before);
    int spaces = 0;
    int words = 0;
    for(int i = 0; i < *sent_before; i++){
      for(int j = 0; j < strlen(text[i]); j++){
        if(text[i][j] == ' '){
          spaces ++;
        }
      }
      words = spaces +1;
      if (words > aver_w){
        for(int p = i; p < *sent_before; p++){
          text[p] = text[p+1];
        }
        i--;
        *sent_before = *sent_before - 1;
      }
      spaces = 0;
      words = 0;
    }
```

```
    return text;
}
//-------<Сортировка по убыванию времени>-------------------------------
-------

char** sortirovka_time(char** text, int* sent_before){
  int times[*sent_before-1][3];
  int total_time[*sent_before-1];
  for(int i = 0; i < *sent_before -1; i++){
    times[i][0] = 0;
    times[i][1] = 0;
    times[i][2] = 0;
    if(strlen(text[i])-1 >= 8){
      for(int j = 0; j < strlen(text[i])-8; j++){
        if((j >0 && isdigit(text[i][j]) && text[i][j-1] == ' ') || (j ==
0 && isdigit(text[i][j]))){
          if(isdigit(text[i][j+1])){
            if(text[i][j+2] == '-'){
              if(isdigit(text[i][j+3])){
                if(isdigit(text[i][j+4])){
                  if(text[i][j+5] == '-'){
                    if(isdigit(text[i][j+6])){
                      if(isdigit(text[i][j+7])&& (text[i][j+8] == ' ' ||
text[i][j+8] == '.'|| text[i][j+8] == ',')){
                        times[i][0] = times[i][0] + (text[i][j+6]-48)*10
+ (text[i][j+7]-48);
                        times[i][1] = times[i][1] + (text[i][j+3]-48)*10
+ (text[i][j+4]-48);
                        times[i][2] = times[i][2] + (text[i][j]-48)*10 +
(text[i][j+1]-48);
                      }
                    }
                  }
                }
              }
            }
          }
        }
```

```
      }
    }
    times[i][1] = times[i][1]+times[i][0]/60;
    times[i][2] = times[i][2]+times[i][1]/60;
    times[i][1] = times[i][1]%60;
    times[i][0] = times[i][0]%60;
    total_time[i] = times[i][2]*10000 + times[i][1]*100 + times[i][0];
  }
  int buf = 0;
  char *sent;
  sent = malloc(10*sizeof(char));
  for(int i = 0; i< *sent_before-1; i++){
    for(int j = i; j< *sent_before-1;j++){
      if(total_time[j] > total_time[i]){
        buf = total_time[i];
        total_time[i] = total_time[j];
        total_time[j] = buf;

        sent = realloc(sent, strlen(text[i]));
        strcpy(sent, text[i]);
        strcpy(text[i], text[j]);
        strcpy(text[j], sent);
      }
    }
  }
  return text;
}

//-------<Замена строчных согласных на заглавные согласные>--------------
-------

char** consonant_up(char** text, int* sent_before){
  char a[20] = {'b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n',
'p', 'r', 's', 't', 'v', 'w', 'x', 'z'};
  for(int i = 0; i < *sent_before -1; i++){
    for(int j = 0; j< strlen(text[i]); j++){
      for(int k = 0; k<20; k++){
        if(tolower(text[i][j]) == a[k]){
```

```
          text[i][j] = toupper(text[i][j]);
        }
      }
    }
  }
  return text;
}

//-------<Output>-------------------------------------------------------
-------

void output(char** text, int* sent_before){
    for(int i = 0; i<*sent_before-1; i++){
      printf("%s", text[i]);
    }
    printf("\n");
}
//-------<Menu>---------------------------------------------------------
-------

void menu(char** text, int* sent_before, int n){
  int k;
  while(1){
    switch(n){
        case 1:
          text = sortirovka_time(text, sent_before);
          output(text, sent_before);
          k = 1;
          break;
        case 2:
          text = words(text, sent_before);
          output(text, sent_before);
          k = 1;
          break;
        case 3:
          text = consonant_up(text, sent_before);
          output(text, sent_before);
          k = 1;
```

```
          break;
        case 4:
          text = delete_less_average_words(text, sent_before);
          output(text, sent_before);
          k = 1;
          break;
        case 5:
          k = 2;
          break;
        default:
          printf("Вы ввели недоступное действие. Введите доступное
действие.\n");
          break;
      }
    if(k == 1){
      printf("\nВведите выбранный номер действия: ");
      scanf("%d", &n);
    }
    if(k == 2){
      break;
    }
  }
}

//-----------------------------------------------------------------------
-------

int main()
{
    char** text;
    int n;
    int sent_before = 0;
    printf("\tЗдравствуйте, дорогой пользователь!\n");
    printf("Вам доступны следующие действия по обработке текста:\n");
    printf("  1.Отсортировать все предложение по уменьшению суммарного
времени указанного в строке.\n");
    printf("  2.Вывести все предложения в которых количество слов больше
2 и меньше 7.\n");
```

```
    printf("  3.Во всем тексте заменить согласные строчные буквы на
заглавные.\n");
    printf("  4.Удалить все предложения в которых количество слов больше
среднего количества слов предложениях.\n");
    printf("  5.Прекращение работы программы.\n");
    printf("Введите ваш текст:\n");
    text = input(&sent_before);
    text = delete_repeat(text, &sent_before);
    printf("\nВведите выбранный номер действия: ");
    scanf("%d", &n);
    menu(text, &sent_before,n);
    for(int j = 0; j < sent_before ; j++){
        free(text[j]);
    }
    free(text);
    return 0;
}
```