

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 Си

Студентка гр. 3382

Копасова К. А.

Преподаватель

Глазунов С.А.

Санкт-Петербург
2023

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студентка Копасова К. А.

Группа 3382

Тема работы: Обработка строк на языке Си

Исходные данные:

Вариант 4.6

Вывод программы должен быть произведен в стандартный поток вывода: stdout.

Ввод данных в программе в стандартный поток ввода: stdin.

В случае использования Makefile название исполняемого файла должно быть: sw.

Важно: первой строкой при запуске программы нужно выводить информацию о варианте курсовой работе и об авторе программы в строго определенном формате:

Course work for option <V>, created by <Name> <Surname>.

Где V – вариант курсовой и Имя и Фамилия, как указано в репозитории группы. Данное предложение должно быть строго первым предложением в выводе программы и является отдельной строкой (заканчивается знаком '\n').

Например:

Course work for option 3.2, created by Ivan Ivanov.

Ввод данных:

После вывода информации о варианте курсовой работе программа ожидает ввода пользователем числа – номера команды:

0 – вывод текста после первичной обязательной обработки (если предусмотрена заданием данного уровня сложности)

1 – вызов функции под номером 1 из списка задания

2 – вызов функции под номером 2 из списка задания

3 – вызов функции под номером 3 из списка задания

4 – вызов функции под номером 4 из списка задания

5 – вывод справки о функциях, которые реализует программа.

Программа не должна выводить никаких строк, пока пользователь не введет число.

В случае вызова справки (опция 5) текст на вход подаваться не должен, во всех остальных случаях после выбора опции должен быть считан текст.

Признаком конца текста считается два подряд идущих символа переноса строки '\n'. После каждой из функций нужно вывести результат работы программы и завершить программу.

В случае ошибки и невозможности выполнить функцию по какой-либо причине, нужно вывести строку:

Error: <причина ошибки>

Каждое предложение должно выводиться в отдельной строке, пустых строк быть не должно. Текст представляет собой предложения, разделенные точкой. Предложения - набор слов, разделенные пробелом или запятой, слова - набор латинских букв и цифр. 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).

Программа должна выполнить одно из введенных пользователем действий и завершить работу:

Распечатать каждое слово и количество его повторений (независимо от регистра) в тексте.

Преобразовать каждое предложение так, что символы в каждом слове шли в обратном порядке.

Удалить предложения, в которых встречается запятая.

Отсортировать предложения по уменьшению значения кода 5 символа предложения. Если 5 символ является разделителем между словами, то брать следующий символ. Если символов в предложении меньше 5, то для этого предложения значение равняется -1.

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.

Содержание пояснительной записки:

Аннотация, введение, описание программы, заключение, список используемых источников, тестирование и исходный код.

Предполагаемый объем пояснительной записки:

Не менее 35 страниц.

Дата выдачи задания: 16.10.2023

Дата сдачи реферата: 11.12.2023

Дата защиты реферата: 15.12.2023

Студентка

Копасова К. А.

Преподаватель

Глазунов С.А.

АННОТАЦИЯ

Программа принимает на вход текст, состоящий из предложений, разделённые точкой. Программа сохраняет данный текст в динамический массив строки, удаляет все повторяющиеся предложения и выполняет одно из введенных пользователем действий: вывод текста после первичной обработки; вызов функции 1 – печать каждого слова и количество его повторений; вызов функции 2 – преобразование предложений так, что символы в каждом слове идут в обратном порядке; вызов функции 3 – удаление предложений с запятой; вызов функции 4 – сортировка предложений по уменьшению значения кода 5 символа предложения; вывод справки о функциях. Программа завершается, после вывода результата одной из пяти функций, описанных выше.

SUMMARY

The program accepts text consisting of sentences separated by a dot as input. The program saves this text into a dynamic string array, deletes all duplicate sentences and performs one of the actions entered by the user: text output after initial processing; function call 1 – printing each word and the number of its repetitions; function call 2 – converting sentences so that the characters in each word go in reverse order; function call 3 – deleting sentences with a comma; function call 4 – sorting sentences to reduce the value of the code 5 of the sentence character; output help about functions. The program ends after the output of one of the five functions described above.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	7
ОСНОВНАЯ ЧАСТЬ	8
Объявление библиотек и постоянных.....	8
Функция main	10
Функция вывода справки	12
Функция ввода и вывода	13
Функция разделения текста на предложения.....	15
Функция освобождения динамической памяти	17
Функция удаления одинаковых предложений.....	18
Функция 1.	19
Функция 2	20
Функция 3	21
Функция 4	22
Заключение	23
Список использованных источников	24
ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ	25
ПРИЛОЖЕНИЕ Б. РЕЗУЛЬТАТ ТЕСТИРОВАНИЯ.....	35

ВВЕДЕНИЕ

Целью данной работы является обобщение полученных знаний за 1 семестр по дисциплине «программирование» о языке программирования Си, а также их применение на практике.

Задачи:

- 1) Реализовать функции ввода, вывода;
- 2) Реализовать функцию удаления одинаковых предложений;
- 3) Реализовать функцию освобождения динамической памяти;
- 4) Реализовать функцию разделения текста на предложения;
- 5) Реализовать функции 1, 2, 3 и 4.

ОСНОВНАЯ ЧАСТЬ

Объявление библиотек и постоянных

Для корректной работы компилятора подключим директивы препроцессора:

```
#define _CRT_SECURE_NO_WARNINGS
```

```
#define _CRT_NONSTDC_NO_DEPRECATED
```

Они используются для подавления предупреждений компилятора. Они отключают предупреждения, связанные с использованием небезопасных функций стандартной библиотеки Си.

Подключим заголовочные файлы стандартных библиотек Си:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <locale.h>
```

```
#include <ctype.h>
```

Они содержат определения функций, констант и типов данных, которые будут использоваться в программе.

Далее подключим pragma-директиву (используется для корректной работы русского языка в VS Studio), которая позволяет установить кодировку UTF-8 для поддержки юникода:

```
#pragma execution_character_set("utf-8")
```

Используем макрос define для определения строки INF_ABOUT_STaW (информация о студенте и его работе) и константы начальной длины текста:

```
#define INF_ABOUT_STaW "Course work for option 4.6, created by Kseniya  
Kopasova\n"
```

```
#define LEN_TEXT 100
```

После программа объявляет все функции, которые определены в дальнейшем коде. Объявление функций важно для того, чтобы компилятор знал о существовании этих функций до их фактического определения. Это

позволяет использовать эти функции в других частях программы до их фактической реализации. Кроме того, это улучшает структурированность и удобство чтения кода, так как разделение объявлений функций от их реализации делает код более организованным и понятным для разработчиков.

Функция main

В самом начале программа выводит ту самую константу INF_ABOUT_STaW – в ней содержится информация о авторе курсовой работы и вариант работы. Далее происходит установка локали для корректной работы кириллицы. Затем пользователь должен ввести переменную func_value, которая будет отвечать за выбор реализуемой функции (для считывания func_value используем функцию библиотеки stdio языка Си scanf()). Далее программа проверяет значение переменной func_value с помощью функции isdigit() - если func_value не является цифрой, то программа должна вывести ошибку о некорректности ввода.

Далее идет блок switch(), который осуществляет выбор различных действий в зависимости от введенного символа. Он содержит несколько случаев:

1) func_value = 0 – программа считывает текст с помощью функции ввода, после делит этот текст на предложения, удаляя одинаковые предложения, и после выводит результат обработки, очищая динамическую память.

2) func_value = 1 - программа считывает текст с помощью функции ввода, после делит этот текст на предложения, удаляя одинаковые предложения, выводит каждое слово и его количество в тексте и очищает динамическую память.

3) func_value = 2 - программа считывает текст с помощью функции ввода, после делит этот текст на предложения, удаляя одинаковые предложения, после выводит предложения, в которых слова «повернуты наоборот», и очищает динамическую память.

4) func_value = 3 - программа считывает текст с помощью функции ввода, после делит этот текст на предложения, удаляя одинаковые предложения, после выводит предложения, в которых нет запятых, и очищает динамическую память.

5) `func_value = 4` – программа считывает текст с помощью функции ввода, после делит этот текст на предложения, удаляя одинаковые предложения, после выводит предложения, отсортированные с пятого символа в порядке уменьшения кода (если пятый символ не является разделителем в предложении), и очищает динамическую память.

6) `func_value = 5` – программа выводит справку о всех функциях программы.

Если значение переменной не соответствует вышеописанным значениям, то программа выводит ошибку о некорректности ввода.

После выполнения соответствующего блока кода, происходит возврат значения 0.

Функция вывода справки

Функция `file_help()` предназначена для вывода справочной информации о функциях, которые реализует программа.

Внутри функции используется функция `printf()`, которая выводит на экран текст, разделенный на несколько строк. Каждая строка текста представляет собой описание определенной функции программы.

Таким образом, при вызове функции `file_help()` на экран будет выведена справочная информация о функциях программы.

Функция ввода и вывода

1. Функция ввода:

Функция ввода `text_input_func` предназначена для считывания подаваемого пользователем текста, который будет записан в двумерный массив `text`.

Создаем переменную `len`, которая будет использоваться для отслеживания текущей длины введенного текста. Далее инициализируем переменную `add_memory = LEN_TEXT`, которая равна константе, определенной через макрос. Определяем переменную `input_char`, которая будет использоваться для считывания символов с помощью функции `getchar()`. Выделяем память под массив символов `text` с начальным размером `add_memory` с помощью функции `malloc`. Затем используется цикл `while()`, в котором считывается каждый символ, введенный с клавиатуры, с помощью функции `getchar()`. Если введен символ новой строки `'\n'`, то проверяется, является ли он вторым введенным символом новой строки `'\n'`. Если условие выполняется, то ввод завершается.

Далее проверяется, не начинается ли текст с символа `'!'`. Если это так, то выводится сообщение об ошибке.

Затем проверяется, не достигла ли длина введенного текста размера выделенной памяти. Если это так, то выделяется дополнительная память с помощью функции `realloc()`.

После этого каждый введенный символ добавляется в массив `text`, а переменная `len` увеличивается на 1.

После завершения ввода текста добавляется завершающий нулевой символ конца строки `'\0'`. Затем проверяется, не заканчивается ли текст символом `'.'`. Если это не так, то добавляется точка в конец текста.

Наконец, функция возвращает указатель на массив `text`, содержащий введенный текст.

2. Функция вывода:

Функция `text_print_func()` предназначена для вывода текста, который был разделен на отдельные строки с помощью функции `split_text_func()`.

На вход функция принимает указатель на массив указателей на строки (`splitted_text`) и количество строк (`output_numb`), которые были разделены из введенного текста.

Далее происходит цикл, в котором каждая строка из массива `splitted_text` поочередно выводится на экран с помощью функции `printf()`.

Таким образом, функция позволяет вывести на экран введенный текст, разделенный на отдельные строки.

Функция разделения текста на предложения

Функция `split_text_func()` предназначена для разделения текста на предложения с использованием определенного разделителя и удаления определенных предложений в зависимости от значения параметра `func_value`.

Внутри функции происходит следующее:

1. Выделение памяти под массив указателей `splitted_text`, который будет содержать разделенные предложения.

2. Инициализация счетчиков `count_sent` (для подсчета предложений), `last_ind` (для отслеживания длины текущего предложения), `deleting_sent` (для флага удаления предложения) и `was_letter_printed` (для отслеживания того, была ли напечатана буква в текущем предложении).

3. Проход по каждому символу входного текста и выполнение следующих действий:

- Если вызвана функция 3 и текущий символ - запятая, устанавливается флаг `deleting_sent`.

- Если текущий символ - разделитель, происходит подсчет предложений, выделение памяти под новое предложение в массиве `splitted_text` и заполнение этого предложения символами из входного текста.

- После завершения обработки текущего предложения, устанавливается `last_ind` в 0 и сбрасывается флаг `was_letter_printed`.

4. После прохода по всем символам входного текста, количество найденных предложений записывается в переменную `output_numb` и вызывается функция `text_deleting_repeat_func()` для удаления повторяющихся предложений.

5. Функция возвращает массив указателей `splitted_text`, содержащий разделенные предложения.

Таким образом, функция `split_text_func()` разделяет входной текст на предложения, учитывая заданный разделитель и условия удаления

определенных предложений, и возвращает массив указателей с разделенными предложениями.

Функция освобождения динамической памяти

Функция `text_free_func()` предназначена для освобождения динамической памяти, выделенной в функции `split_text_func()` для массива указателей `splitted_text` и входного текста.

Внутри функции происходит следующее:

1. Проход по каждому элементу массива указателей `splitted_text` с помощью цикла `for`.
2. Для каждого элемента массива вызывается функция `free()` для освобождения памяти, выделенной под каждое предложение.
3. После освобождения памяти для всех предложений освобождается сам массив указателей `splitted_text` с помощью функции `free()`.
4. Также освобождается память, выделенная под входной текст с помощью функции `free()`.

Таким образом, функция `text_free_func()` обеспечивает корректное освобождение всей выделенной динамической памяти после завершения работы функции `split_text_func()`.

Функция удаления одинаковых предложений

Функция `text_deleting_repeat_func()` предназначена для удаления одинаковых предложений из массива указателей `splitted_text`.

Внутри функции происходит следующее:

1. Проход по каждому элементу массива указателей `splitted_text` с помощью вложенных циклов `for`.
2. Для каждого элемента `i` происходит проверка на равенство с каждым элементом `j`, начиная с `i+1`.
3. Если предложения равны (с помощью функции `strcmp()`), то освобождается память, выделенная под предложение `j` с помощью функции `free()`.
4. Затем происходит сдвиг всех элементов массива, начиная с `j+1`, на одну позицию влево, чтобы удалить пустую ячейку в массиве.
5. Уменьшается значение переменной `output_numb`, чтобы отразить факт удаления дубликата предложения.
6. После завершения работы функции массив `splitted_text` будет содержать только уникальные предложения, а переменная `output_numb` будет содержать актуальное количество предложений в массиве.

Функция 1.

Функция 1 представляет из себя несколько подфункций и структуру для удобства программы.

1) Функция `compare()` используется в качестве функции сравнения для сортировки массива структур `WordCount`. Она принимает указатели на две структуры и сравнивает их слова с помощью функции `strcmp()`, возвращая результат сравнения.

2) Функция `toLower()` принимает строку и приводит все символы к нижнему регистру с помощью цикла `for` и функции `tolower()`.

3) Функция `print_word_and_number_repetitions()` принимает текст в качестве аргумента. Сначала приводит текст к нижнему регистру с помощью вызова `toLower()`. Затем создается массив структур `WordCount`, выделяется память под 100 элементов. Затем происходит разделение текста на слова с помощью функции `strtok()` и подсчет количества повторений каждого слова в тексте. Если слово уже встречалось, увеличивается его счетчик, иначе добавляется новая структура `WordCount` в массив.

После этого массив сортируется с помощью функции `qsort()`, используя функцию `compare()` в качестве функции сравнения.

Затем происходит вывод отсортированного массива структур `WordCount`, печатая каждое слово и количество его повторений. После этого освобождается память, выделенная под каждое слово, и затем освобождается память, выделенная под массив структур `WordCount`.

Функция 2

Функция `text_reverse()` принимает строку `text` в качестве аргумента и возвращает указатель на измененную строку.

Сначала функция определяет длину строки `text` с помощью функции `strlen()`. Затем она инициализирует переменные `start` и `fl`, которые будут использоваться для определения начала и конца слова в строке.

Далее происходит цикл `for`, который проходит по каждому символу в строке. Если символ является буквой или цифрой (с помощью функции `isalnum()`), и переменная `fl` равна 0, это означает начало нового слова. Тогда переменная `start` устанавливается равной текущему индексу `i`, и переменная `fl` устанавливается в 1.

Затем, если текущий символ является знаком препинания (',' или '.'), или пробелом, и предыдущий символ был буквой или цифрой, это означает конец слова. Тогда переменная `stop` устанавливается равной `i - 1`, и происходит цикл `for`, который разворачивает слово между индексами `start` и `stop`, меняя местами символы.

После завершения цикла возвращается указатель на измененную строку `text`.

Таким образом, функция `text_reverse()` разворачивает каждое слово в строке, оставляя пунктуацию и пробелы на своих местах.

Функция 3

Данная функция удаления предложений, в которых есть запятая, реализована в функции `split_text_func()`. Если предложение имеет хотя бы одну запятую, то программа его не записывает. Подробнее про функцию `split_text_func` написано на ст. 16-17.

Функция 4

Функция 4 состоит из подфункции `bubble_sort()` и основной функции `text_func_four()`:

1) Функция `bubble_sort` – она принимает массив строк `splitted_text`, количество строк в массиве `output_numb` и индекс `index`. Она использует алгоритм сортировки пузырьком для сортировки символов в каждой строке массива `splitted_text`.

Алгоритм сортировки пузырьком работает следующим образом: проходим по каждому элементу массива и сравниваем его с каждым последующим элементом. Если текущий элемент больше следующего, меняем их местами. Повторяем этот процесс до тех пор, пока массив не будет отсортирован.

2) Функция `text_func_four` – она также принимает массив строк `splitted_text` и количество строк в массиве `output_numb`. Она проходит по каждой строке массива и проверяет, содержит ли она букву или цифру в позиции пятого символа. Если предложение содержит пятый символ и длина предложения больше 5, то вызывается функция `bubble_sort` для сортировки символов в строке начиная с позиции `index = 4` (как раз пятый символ строки). Если не содержит, то происходит поиск первой буквы или цифры после позиции `index = 4` и вызывается функция `bubble_sort` для сортировки символов начиная с этой позиции.

После сортировки функция выводит отсортированные строки из массива `splitted_text`, если их длина больше 5, иначе выводится -1.

Заключение

В ходе курсовой работы я обобщила знания по дисциплине «программирование» за 1 семестр по языку программирования Си и закрепила их на практике.

Была написана программа, которая успешно отрабатывает текст произвольной длины и выполняет все функции, которое требует техническое задание.

Список использованных источников

1. Онлайн-курс «Программирование на Си. Практические задания. Первый семестр» // URL: <https://e.moevm.info/>
2. Д. Ритчи, Б. Керниган Язык программирования Си: 3-е издание.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл: main.c

```
#define _CRT_SECURE_NO_WARNINGS
#define _CRT_NONSTDC_NO_DEPRECATED
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include <ctype.h>
#pragma execution_character_set("utf-8")
#define INF_ABOUT_STaW "Course work for option 4.6, created by Kseniya Kopasova\n"
#define LEN_TEXT 100

//объявляем все функции
void file_help();
void text_print_func(char** splitted_text, int output_numb);
char* text_input_func();
char** split_text_func(char* text, char* splitter, int* output_numb, int func_value);
void text_free_func(char** splitted_text, char* text, int output_numb);
void text_deleting_repeat_func(char** splitted_text, int* output_numb);
void print_word_and_number_repetitions(char* text);
//int numb_word(char** splitted_text, int output_numb);
char* text_reverse(char* text);
void toLower(char* text);
void bubble_sort(char** splitted_text, int output_numb, int index);
void text_func_four(char** splitted_text, int output_numb);

int main() {
    printf(INF_ABOUT_STaW);

    ///чтобы компилятор воспринимал кириллицу
    setlocale(LC_ALL, "ru_RU.UTF-8");
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    ///
    char func_value = 0; //значение функции
    scanf("%c", &func_value);
    getchar();

    if (!isdigit(func_value)) {
        printf("Error: Некорректный ввод.\n");
    }
}
```

```

int output_numb = 0;
char** splitted_text;
char* text;

//////////

switch (func_value) {
case '0': //выводим текст после первичной обработки
    //ввод и первичная обработка
    text = text_input_func();
    splitted_text = split_text_func(text, ".", &output_numb,
func_value);

    //выводим текст после первичной обработки и освобождаем
память
    text_print_func(splitted_text, output_numb);
    text_free_func(splitted_text, text, output_numb);

    break;

case '1': //выводим функцию под номером 1
    //ввод и первичная обработка
    text = text_input_func();
    splitted_text = split_text_func(text, ".", &output_numb,
func_value);

    //выводим текст после первичной обработки и освобождаем
память
    print_word_and_number_repetitions(text);
    text_free_func(splitted_text, text, output_numb);

    break;

case '2': //выводим функцию под номером 2

    //ввод и обработка
    text = text_input_func();
    splitted_text = split_text_func(text_reverse(text), ".",
&output_numb, func_value);

    //вывод и освобождение памяти
    text_print_func(splitted_text, output_numb);
    text_free_func(splitted_text, text, output_numb);

    break;

case '3': //выводим функцию под номером 3

```

```

        //обработка и ввод
        text = text_input_func();
        splitted_text = split_text_func(text, ".", &output_numb,
func_value);

        //вывод и освобождение памяти
        text_print_func(splitted_text, output_numb);
        text_free_func(splitted_text, text, output_numb);

        break;

    case '4': //выводим функцию под номером 4
        //ввод и обработка
        text = text_input_func();
        splitted_text = split_text_func(text, ".", &output_numb,
func_value);

        //вывод и освобождение памяти
        text_func_four(splitted_text, output_numb);
        text_free_func(splitted_text, text, output_numb);
        break;

    case '5': //выводим справки о всех функциях

        file_help();
        break;

    default:

        printf("Error: Неправильно введена функция.\n");
        break;
}

return 0;
}

//функция для вывода справки (func_value=5)
void file_help() {
    printf("%s\n%s\n%s\n%s\n%s\n%s\n%s\n",
        "Справка:",
        "0 - вывод текста после первичной обязательной обработки;",
        "1 - распечатать каждое слово и количество его повторений
в тексте;",
        "2 - преобразовать каждое предложение так, что символы в
каждом слове шли в обратном порядке;",
        "3 - удалить предложения, в которых встречается запятая;",
        "4 - отсортировать предложения по уменьшению значения кода
5 символа предложения;",

```

```

        "5 - вывод справки о функциях, которые реализует
программа.");
    }

//принимаем текст
char* text_input_func() {

    int len = 0;
    int add_memory = LEN_TEXT;
    char input_char;
    char* text = malloc(sizeof(char) * add_memory); //выделяем
память на массив с указателями

    //динамическое выделение памяти под исходный текст
    while ((input_char = getchar())) {
        if ((input_char == '\n')) {
            if ((len > 0) && (text[len - 1] == '\n')) {
                break;
            }
        }

        if (text[0] == '.') {
            printf("Error: Ошибка ввода.\n");
        }
        if (len >= add_memory && text != NULL) {
            add_memory += LEN_TEXT;
            text = realloc(text, sizeof(char) * add_memory);
        }
        text[len++] = input_char;
    }

    text[len] = '\0';

    if (text[strlen(text) - 2] != '.') {
        text[strlen(text) - 2] = '.';
    }
    return text;
}

//выводим текст
void text_print_func(char** splitted_text, int output_numb) {
    for (int i = 0; i < output_numb; i++) {
        printf("%s\n", splitted_text[i]);
    }
}

//функция освобождения динамической памяти
void text_free_func(char** splitted_text, char* text, int
output_numb) {

```

```

        for (int i = 0; i < output_numb; i++) {
            free(splitted_text[i]);
        }
        free(splitted_text);
        free(text);
    }

//функция удаления одинаковых предложений
void text_deleting_repeat_func(char** splitted_text, int*
output_numb) {
    for (int i = 0; (i < (*output_numb)) && (splitted_text[i]); i++)
    {
        for (int j = i + 1; (j < (*output_numb)) &&
(splitted_text[j]); j++) {
            if (strcmp(splitted_text[i], splitted_text[j]) == 0)
            {
                free(splitted_text[j]);
                for (int k = j; k < (*output_numb); k++) {
                    splitted_text[k] = splitted_text[k + 1];
                }
                (*output_numb)--;
                j--;
            }
        }
    }
}

//функция разделения предложений и удаления предложений с запятыми
char** split_text_func(char* text, char* splitter, int* output_numb,
int func_value) {
    char** splitted_text = malloc(sizeof(char*));

    int count_sent = 0; //счетчик предложений
    int last_ind = 0;
    int deleting_sent = 0; //флаг удаления предложения
    int was_letter_printed = 0;

    for (int i = 0; i < (int)strlen(text); i++) {
        //если вызвана функция 3, то удаляем предложение с запятой
        if ((func_value == '3') && (text[i] == ',')) {
            deleting_sent = 1;
        }

        // считаем предложения и разделяем текст на предложения
        if ( strchr(splitter, text[i])) {

            if (deleting_sent == 1) {
                deleting_sent = 0;
                last_ind = 0;
            }
        }
    }
}

```

```

        continue;
    }

    count_sent++;
    if (splitted_text != NULL && count_sent != 0 &&
last_ind != 0) {
        splitted_text = realloc(splitted_text,
count_sent * sizeof(char*));
        splitted_text[count_sent - 1] = malloc((last_ind
+ 2) * sizeof(char));
    }
    int chr_temp = 0;
    for (int j = last_ind; j >= 0; j--) {
        ////////////////пропуск запятой и точки в начале
предложения

        if ((!isspace(text[i - j])) && (text[i - j] !=
', ')) {
            was_letter_printed = 1;
        }
        if ((!was_letter_printed) && (isspace(text[i -
j])) || (text[i - j] == ', ')) {
            continue;
        }
        splitted_text[count_sent - 1][chr_temp++] =
text[i - j];
    }

    while (isspace(splitted_text[count_sent - 1][chr_temp
- 1])) {
        chr_temp--;
    }

    splitted_text[count_sent - 1][chr_temp] = '\\0';
    last_ind = 0;
    continue;
}
last_ind++;
was_letter_printed = 0;
}
*output_numb = count_sent;
text_deleting_repeat_func(splitted_text, output_numb);
return splitted_text;
}

//считаем количество слов
/*
int numb_word(char** splitted_text, int output_numb) {
    int word_count = 0;

```

```

        for (int i = 0; i < output_numb; i++) {
            for (int j = 0; j < ((int)strlen(splitted_text[i])+1); j++)
            {
                if (((splitted_text[i][j] == ' ' ||
(splitted_text[i][j] == ',') || (splitted_text[i][j] == '.')) &&
((splitted_text[i][j - 1] != ' ') && (splitted_text[i][j - 1] != ',')
&& (splitted_text[i][j - 1] != '.'))) {
                    word_count++;
                }
            }
        }
        return word_count;
    }
*/

//все для функции 1
struct WordCount {
    char* word;
    int count;
};

int compare(const void* a, const void* b) {
    struct WordCount* f = (struct WordCount*)a;
    struct WordCount* s = (struct WordCount*)b;
    return strcmp(f->word, s->word);
}

void toLower(char* text) {
    for (int i = 0; text[i]; i++) {
        text[i] = tolower(text[i]);
    }
}

void print_word_and_number_repetitions(char* text) {
    toLower(text);
    struct WordCount* word_count = malloc(100 * sizeof(struct
WordCount));
    int capacity = 100;
    int count = 0;
    int len_str = strlen(text);
    if (text[len_str - 1] == '\n') {
        text[len_str - 1] = '\0';
    }
    char* temp_word = strtok(text, " ,.");
    while (temp_word != NULL) {
        int fl = 0;
        for (int i = 0; i < count; i++) {
            if (strcmp(word_count[i].word, temp_word) == 0) {
                fl = 1;
            }
        }
        if (fl == 0) {
            word_count[count].word = temp_word;
            count++;
        }
        temp_word = strtok(NULL, " ,.");
    }
}

```

```

        word_count[i].count++;
        break;
    }
}
if (!fl) {
    if (count == capacity) {
        capacity *= 2;
        word_count = realloc(word_count, capacity *
sizeof(struct WordCount));
    }
    word_count[count].word = malloc(strlen(temp_word) +
1);

    strcpy(word_count[count].word, temp_word);
    word_count[count].count = 1;
    count++;
}
temp_word = strtok(NULL, " ,.");
}

qsort(word_count, count, sizeof(struct WordCount), compare);

for (int i = 0; i < count; i++) {
    printf("%s      -      %d\n",      word_count[i].word,
word_count[i].count);
    free(word_count[i].word);
}
free(word_count);
}

//функция 2 преворачиваем слова
char* text_reverse(char* text) {
    int len_str = strlen(text);
    int start = 0;
    int fl = 0;
    for (int i = 0; i <= (len_str); i++) {
        if ((fl == 0) && (isalnum(text[i]))) {
            start = i;
            fl = 1;
        }
        if (((text[i] == ',') || (text[i] == ' ') || (text[i] ==
'.')) && (isalnum(text[i - 1]))) {
            int stop = i - 1;
            for (int j = start, k = stop; j < k; j++, k--) {
                char temp = text[j];
                text[j] = text[k];
                text[k] = temp;
            }
            fl = 0;
        }
    }
}

```



```

    }
    return text;
}

//bubble sort для 4 функции
void bubble_sort(char** splitted_text, int output_numb, int index) {
    char temp;

    for (int k = 0; k < output_numb; k++) {
        int len_sent = strlen(splitted_text[k]);
        for (int i = index; i < len_sent + (index - 1); i++) {
            for (int j = index; j < len_sent - i + (index - 1);
j++) {
                if ((splitted_text[k][j] < splitted_text[k][j +
1])) {
                    temp = splitted_text[k][j];
                    splitted_text[k][j] = splitted_text[k][j +
1];
                    splitted_text[k][j + 1] = temp;
                }
            }
        }
    }
}

//функция 4
void text_func_four(char** splitted_text, int output_numb) {
    for (int i = 0; i < output_numb; i++) {
        int len_sent = strlen(splitted_text[i]);
        int index = 4;
        if (isalnum(splitted_text[i][4]) && ((len_sent - 1) >= 5))
        {
            index = 4;
            bubble_sort(splitted_text, output_numb, index);
            break;
        }
        else {
            for (int l = 0; l < len_sent-4; l++) {
                if (isalnum(splitted_text[i][index+1])) {
                    bubble_sort(splitted_text, output_numb,
index+1);
                    break;
                }
            }
        }
    }
}

```

```
    for (int i = 0; i < output_num; i++) {  
        if (strlen(splitted_text[i]) - 1 < 5) {  
            printf("-1\n");  
        }  
        else {  
            printf("%s\n", splitted_text[i]);  
        }  
    }  
}
```

ПРИЛОЖЕНИЕ Б

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

1. Вывод информации о курсовой работе:

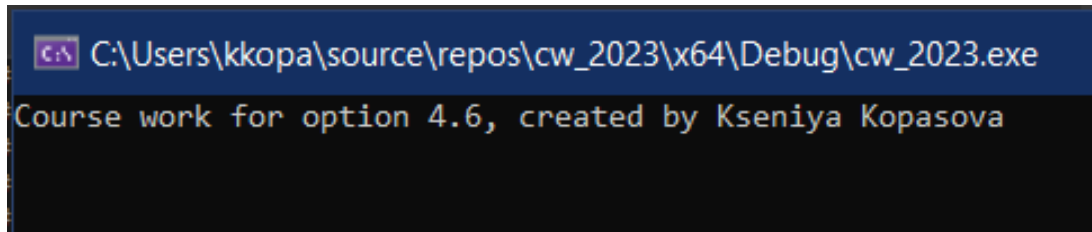


Рисунок 1.

2. Проверка программы на корректное считывание и первоначальную обработку текста:

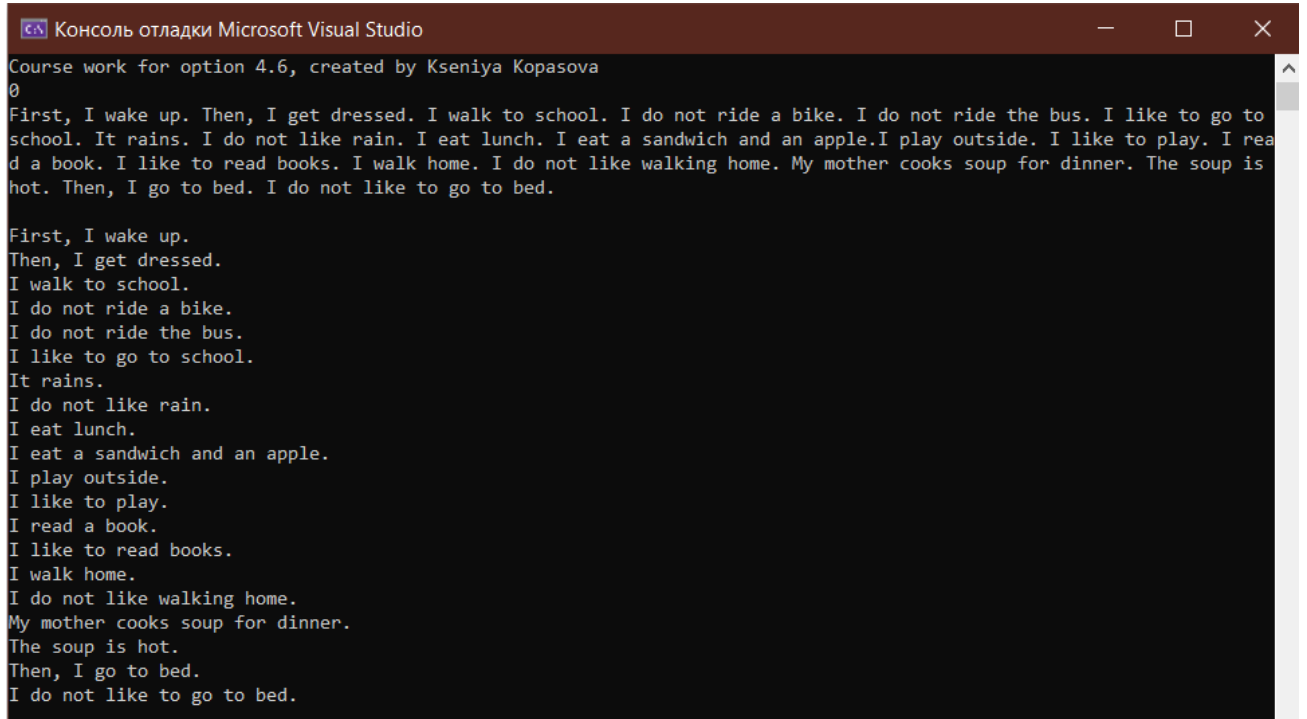


Рисунок 2.1.

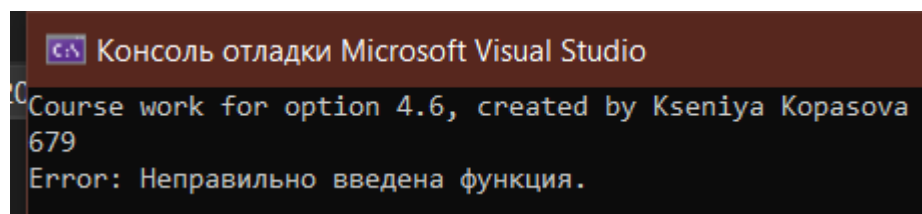


Рисунок 2.2.

```
Консоль отладки Microsoft Visual Studio
Course work for option 4.6, created by Kseniya Kopasova
0
I love this world. I Love this world. I LOVE THIS WORLD. I LOVE this world.

I love this world.

C:\Users\kkopa\source\repos\cw_2023\x64\Debug\cw_2023.exe (процесс 32328) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...
```

Рисунок 2.3.

3. Проверка корректной работы первой функции:

```
Консоль отладки Microsoft Visual Studio
Course work for option 4.6, created by Kseniya Kopasova
1
school Let me tell you about my best friend. His name is Yuriy. We have We we WE known each other for ages. We live in the same town and went to the same school.

about - 1
ages - 1
and - 1
best - 1
each - 1
for - 1
friend - 1
have - 1
his - 1
in - 1
is - 1
known - 1
let - 1
live - 1
me - 1
my - 1
name - 1
other - 1
same - 2
school - 2
tell - 1
the - 2
to - 1
town - 1
we - 5
went - 1
you - 1
yuriy - 1
```

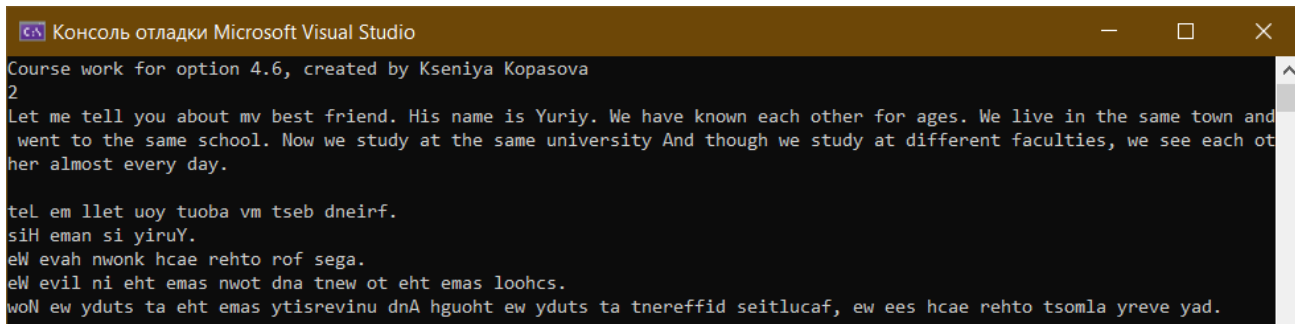
Рисунок 3.1.

```
Консоль отладки Microsoft Visual Studio
Course work for option 4.6, created by Kseniya Kopasova
1
hdb hdb hdb, a a a a A A a a, dhfbshd jsdf. sd sd sd, lka.

a - 9
dhfbshd - 1
hdb - 3
jsdf - 1
lka - 1
sd - 3
```

Рисунок 3.2.

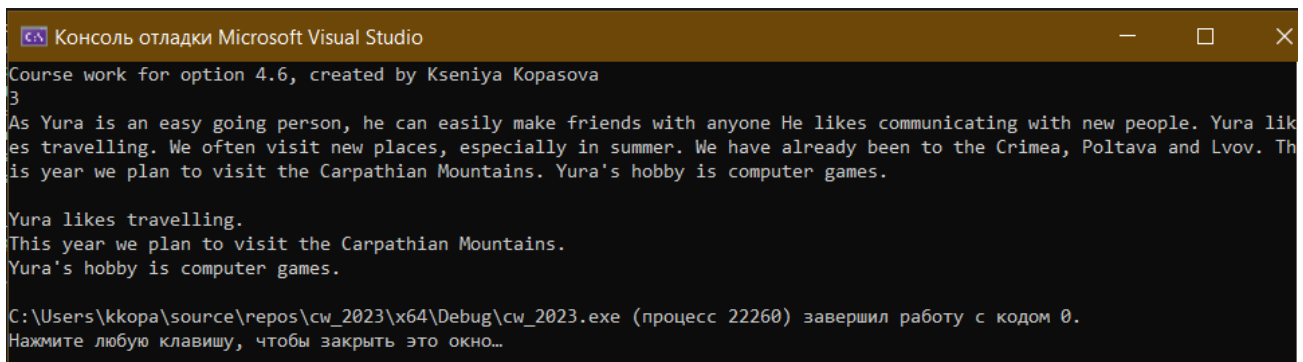
4. Проверка корректной работы второй функции:



```
Консоль отладки Microsoft Visual Studio
Course work for option 4.6, created by Kseniya Kopasova
2
Let me tell you about my best friend. His name is Yuriy. We have known each other for ages. We live in the same town and
went to the same school. Now we study at the same university And though we study at different faculties, we see each ot
her almost every day.
tel em llet uoy tuoba vm tseb dneirf.
siH eman si yiruY.
eW evah nwonk hcae rehto rof sega.
eW evil ni eht emas nwot dna tnew ot eht emas loohcs.
woN ew yduts ta eht emas ytisrevinu dnA hguoht ew yduts ta tnereffid seitlucaf, ew ees hcae rehto tsomla yreve yad.
```

Рисунок 4.

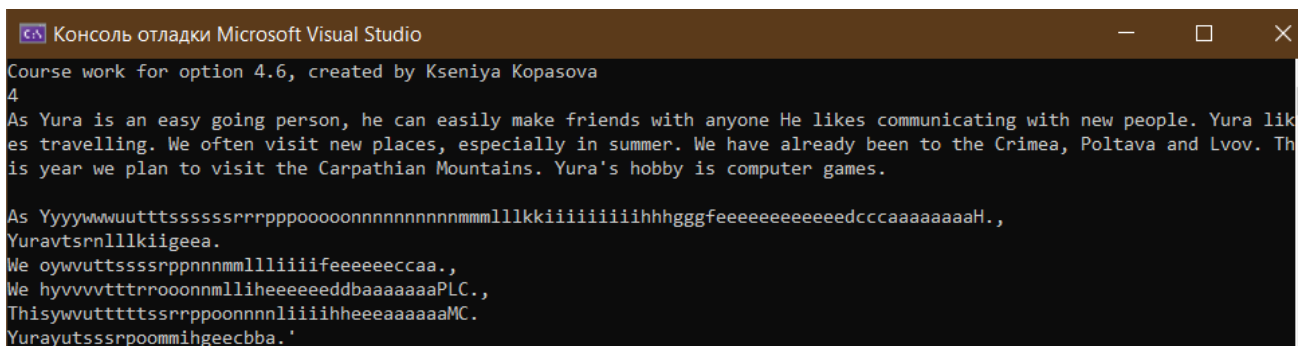
5. Проверка корректной работы третьей функции:



```
Консоль отладки Microsoft Visual Studio
Course work for option 4.6, created by Kseniya Kopasova
3
As Yura is an easy going person, he can easily make friends with anyone He likes communicating with new people. Yura lik
es travelling. We often visit new places, especially in summer. We have already been to the Crimea, Poltava and Lvov. Th
is year we plan to visit the Carpathian Mountains. Yura's hobby is computer games.
Yura likes travelling.
This year we plan to visit the Carpathian Mountains.
Yura's hobby is computer games.
C:\Users\kkopa\source\repos\cw_2023\x64\Debug\cw_2023.exe (процесс 22260) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...
```

Рисунок 5.

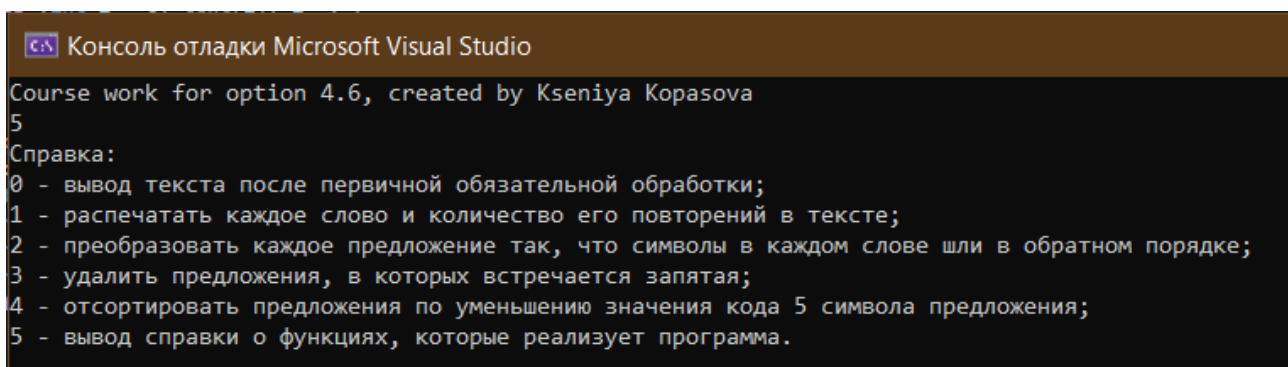
6. Проверка корректной работы четвертой функции:



```
Консоль отладки Microsoft Visual Studio
Course work for option 4.6, created by Kseniya Kopasova
4
As Yura is an easy going person, he can easily make friends with anyone He likes communicating with new people. Yura lik
es travelling. We often visit new places, especially in summer. We have already been to the Crimea, Poltava and Lvov. Th
is year we plan to visit the Carpathian Mountains. Yura's hobby is computer games.
As YyyywwwuuttssssrrpppooooonnnnnnnmmlllkkiiiiiiihhgggfeeeeeeeedcccaaaaaaaH.,
Yuravtsrnl1lkiigee.
We oywvuttsssrppnnmmllliiifeeeeeeccaa.,
We hyvvvtttrrooonnmlliheeeeeeaddbaaaaaaPLC.,
ThisyvvuttsssrppoonnnlliiihheeeaaaaaaMC.
Yurayutsssrpoommihgeecbba.'
```

Рисунок 6.

7. Проверка корректной работы пятой функции:



```
Консоль отладки Microsoft Visual Studio
Course work for option 4.6, created by Kseniya Kopasova
5
Справка:
0 - вывод текста после первичной обязательной обработки;
1 - распечатать каждое слово и количество его повторений в тексте;
2 - преобразовать каждое предложение так, что символы в каждом слове шли в обратном порядке;
3 - удалить предложения, в которых встречается запятая;
4 - отсортировать предложения по уменьшению значения кода 5 символа предложения;
5 - вывод справки о функциях, которые реализует программа.
```

Рисунок 7.