**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

**Курсовая работа**

**по дисциплине "Программирование"**

**Тема: Обработка текстовой информации**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3311 |  | Шарпинский Д. А. |
| Преподаватель |  | Хахаев И. А. |

Санкт-Петербург

2023

**Введение**

Цель работы:

Законченное поэтапное решение содержательной задачи (постановка задачи, спецификация, выбор структур данных и разработка алгоритма, программная реализация, тестирование).

Вариант 65:

Задан текст, содержащий произвольное количество строк, в которых отдельные слова могут разделяться одним или несколькими пробелами и знаками пунктуации (перенос слов с одной строки на другую не используется). Сформировать новый текст, который является результатом следующего преобразования исходного текста: заменить в каждой строке заданное слово на другое слово.

**Постановка задачи и описание решения**

Для решения задачи необходимо осуществить:

1. Возможность ввода многострочного текста (через файл и с клавиатуры), слова, которое будет заменяться, и слова, на которое будет происходить замена.
2. Валидацию слова для замены.
3. Валидацию слова, на которое будет происходить замена.
4. Алгоритм для замены слова.
5. Ввод многострочного текста с клавиатуры осуществляется следующим образом:

До тех пор, пока пользователь не введёт пустую строку, ввод продолжается. Запись текста происходит в строку длиною не более 4096 символов.

Каждая отдельная строка записывается в переменную temp с помощью функции fgets, после чего объединяется с уже записанным текстом в переменной text с помощью strcat.

1. Ввод многострочного текста через файл осуществляется следующим образом:

Сначала у пользователя спрашивается имя файла, затем происходит

попытка открытия файла с таким именем. Если она оказалась успешной, то далее до тех пор, пока объем текста не превысит 4096 символов, происходит запись текста аналогично тому, как это было при вводе с клавиатуры.

1. Валидация слова для замены происходит по следующим критериям:
   1. Слово не должно начинаться с символов разделителей.
   2. Слово не должно быть пустым.

Так, строка, состоящая только из пробелов (или начинающаяся с них), не будет считаться валидной.

1. Валидация слова, на которое будет происходить замена, происходит следующим образом:
   1. Если слово начинается с пробела или символа переноса строки – оно автоматически заменяется на пустую строку.
   2. Если слово содержит в себе пробелы, то оно будет обрезано до первого из них.

4. Алгоритм для замены слова реализован в функции replaceWord:

Функция принимает 4 аргумента (все аргументы –строки):

1. исходная строка src
2. старое слово - строка oldWord
3. новое слово - строка newWord
4. строка dest, в которую будет записан результат.

Внутри функция также имеет 4 переменные:

1. pos – позиция вхождения oldWord в src,
2. len – длина результата (изначально 0),
3. oldLen – длина старого слова (высчитывается с помощью функции strlen),
4. firstSrc – т. к. src в процессе работы функции может изменяться, то необходимо запомнить его начальное значение.

Далее в цикле, пока в строке src есть хотя бы одно вхождение oldWord (проверка происходит с помощью функции strstr, которая записывает в pos указатель на вхождение oldWord в src, то есть условием выхода из цикла является pos == NULL), происходит обработка src и dest.

Внутри цикла идёт ветвление, в зависимости от того, является ли найденное вхождение частью другого слова (проверка осуществляется с помощью просмотра символа до вхождения, если слово в середине строки, и символа после вхождения, то есть на позиции pos + oldLen - предполагаемом конце слова: если оба этих символа являются разделителями, то программа нашла необходимое для замены слово, в противном случае – это слово не подходит, оно является частью другого слова. Например, при oldWord == world и вхождении theworldwide, результатом сравнения будет ложь, то есть 0).

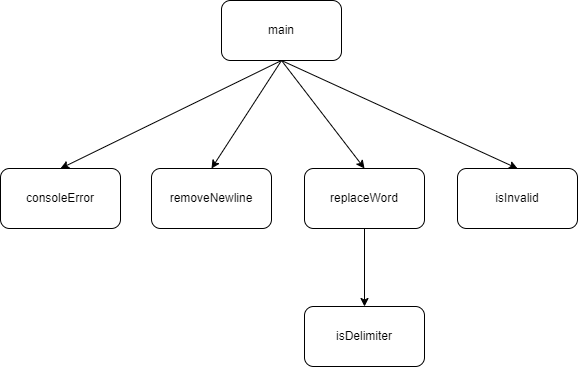
В случае, если найденное вхождение не является частью другого слова, сначала в строку dest дописывается часть строки src до вхождения oldWord с помощью функции strncpy: первым аргументом передаётся указатель dest и длина len (таким образом, запись будет осуществляться в конец строки), вторым src, третьим – разность pos и src. Далее обновляется длина len строки dest: прибавляется разность pos и src. Затем в строку dest дописывается newWord (с помощью функции strcpy), к длине len прибавляется длина newWord, а указатель src сдвигается на pos + oldLen. Таким образом, мы добавили в dest всё, что было до oldWord, после чего дописали newWord, а затем сдвинули src на позицию после текущего вхождения oldWord.

В случае, если найденное вхождение является частью другого слова,

осуществляется копирование в строку dest строки src до позиции pos - src + 1, к длине len прибавляем pos - src + 1, указатель src сдвигаем на pos + 1. Прибавленная единица играет важную роль. Она позволяется избежать бесконечного цикла, который бы образовался, останься указатель src на позиции pos: strstr каждый раз возвращала бы указатель на текущее вхождение.

После выполнения цикла в строку dest дописывается остаток src с помощью функции strcpy.

В конце успешного выполнения программы пользователю выводится введённый им текст (или считанный из файла) и текст, полученный в результате работы программы.

**Структура вызовов функций**

**Описание функций и переменных**

**MAX\_LENGTH – константа типа int. Предназначена для выделения памяти.**

**Функция main**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | text | char | Введённый текст |
| 2 | oldWord | char | Слово для замены |
| 3 | newWord | char | Слово, на которое будет происходить замена |
| 4 | result | char | Обработанный текст |
| 5 | temp | char | Временная переменная, необходимая для записи исходного текста |
| 6 | globalErrorFlag | int | Флаг критической ошибки, при которой нельзя продолжать выполнение программы |
| 7 | inputFlag | int | Флаг окончания ввода |

**Функция isDelimiter – проверяет, является ли символ разделителем.**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | c | char | Символ для проверки |

**Функция isInvalid – проверяет, является ли первый символ строки нуль-теминатором.**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | str | char\* | Строка для проверки |

**Функция consoleError – функция для вывода в консоль сообщения об ошибке.**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | message | char\* | Строка для выведения в консоль |

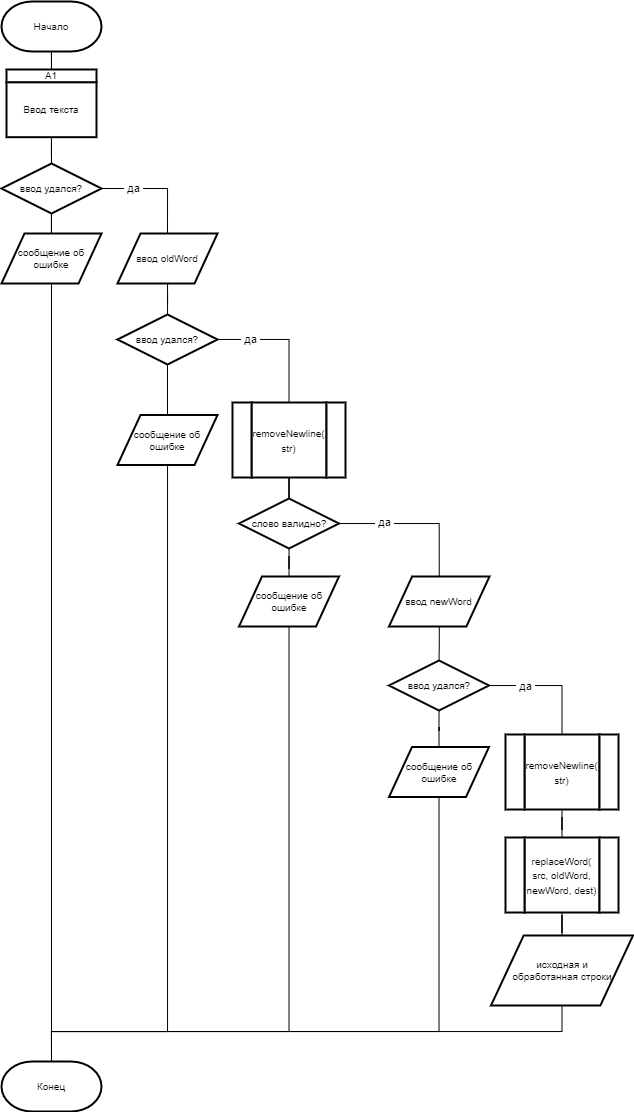
**Функция removeNewline – функция для удаления из строки символа переноса строки. Также обрубает строку на первом найденном пробеле.**

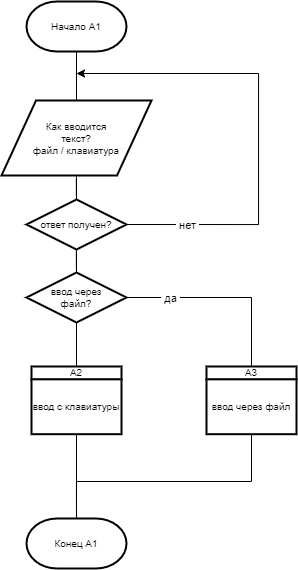
|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | str | char\* | Строка для обработки |

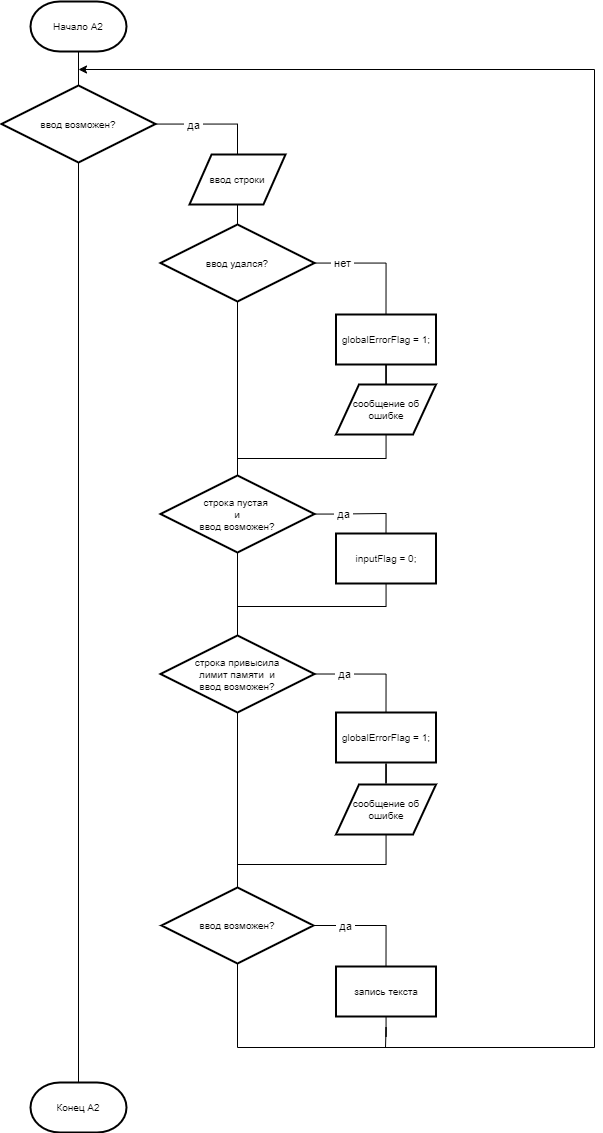
**Функция replaceWord – функция для замены одного слова в строке на другое.**

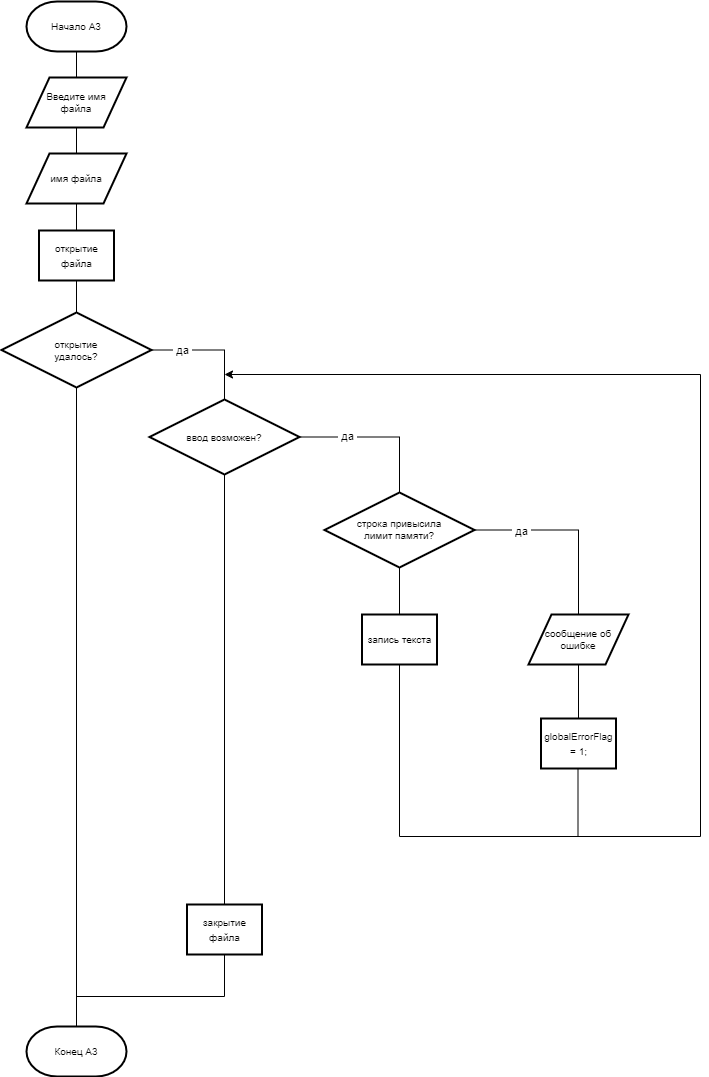
|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменной | Тип | Назначение |
| 1 | src | char\* | Текст для обработки |
| 2 | oldWord | char\* | Слово для замены |
| 3 | newWord | char\* | Слово, на которое будет происходить замена |
| 4 | dest | char\* | Строка, куда записывается обработанный текст |
| 5 | pos | char\* | Позиция вхождения слова для замены в исходном тексте |
| 6 | len | int | Длина обработанного текста |
| 7 | oldLen | int | Длина старого слова |
| 8 | firstSrc | char\* | Начальное значение указателя src |

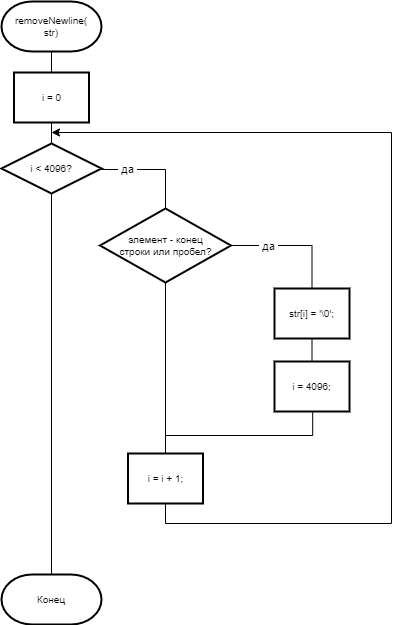
**Схема алгоритма**

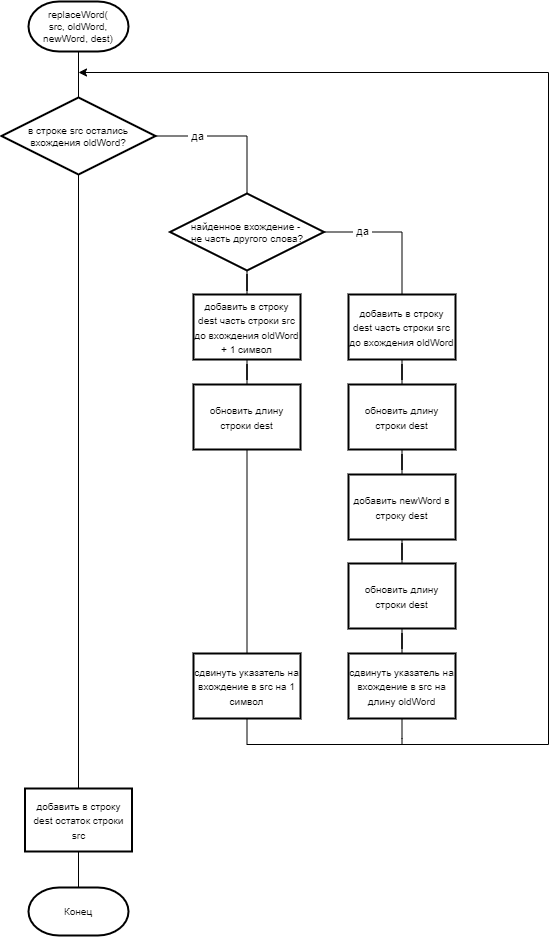
****

****

****

****

****

****

**Контрольные примеры**

**Пример 1:**

Исходный текст:

hello world

worldwide

World

Слово, которое будет заменяться:

world

Слово, на которое будет происходить замена:

word

Обработанный текст:

hello word

worldwide

World

**Пример 2:**

Исходный текст:

wo

wow

wwo

wo

Слово, которое будет заменяться:

wo

Слово, на которое будет происходить замена:

wa

Обработанный текст:

wa

wow

wwo

wa

**Пример 3:**

Исходный текст:

wwwww

wwww

www

ww

w

Слово, которое будет заменяться:

ww

Слово, на которое будет происходить замена:

u

Обработанный текст:

wwwww

wwww

www

u

w

**Пример 4:**

Исходный текст:

1 two 3 four

Слово, которое будет заменяться:

two

Слово, на которое будет происходить замена:

Обработанный текст:

1 3 four

**Пример 5:**

Исходный текст:

example

Слово, которое будет заменяться:

)))

Error: invalid word

**Текст программы**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX\_LENGTH 4096

/\*\* @brief Checking if a character is a delimiter.

@param Character.

@return 1 if character is delimiter else 0.

\*/

int isDelimiter(char c);

/\*\* @brief Checking if a string is invalid (i. e. starts with '\0').

@param Character.

@return 1 string is invalid else 0.

\*/

int isInvalid(const char \*str);

/\*\* @brief Prints error message in console.

@param String message.

@return Void.

\*/

void consoleError(const char \*message);

/\*\* @brief Removes '\n' character from string str. If the string has spaces, it will be cut off to the first of them.

@param String str.

@return Void.

\*/

void removeNewline(char \*str);

/\*\* @brief Replaces oldWord in src string with a newWord and puts the result

in the dest string.

@param String src.

@param String oldWord.

@param String newWord.

@param String dest.

@return Void.

\*/

void replaceWord(char \*src, char \*oldWord, char \*newWord, char \*dest);

/\*\* @brief Clearing console

@return Void.

\*/

void clearConsole();

int main() {

char text[MAX\_LENGTH] = {0}, oldWord[MAX\_LENGTH], newWord[MAX\_LENGTH], result[MAX\_LENGTH] = {0}, temp[MAX\_LENGTH], answer[MAX\_LENGTH];

FILE \*file;

int globalErrorFlag = 0, inputFlag = 1;

do {

printf("how do you want to enter the data (file / keyboard): ");

scanf("%s", answer);

getchar();

} while (strcmp(answer, "keyboard") != 0 && strcmp(answer, "file") != 0 && strcmp(answer, "k") != 0 && strcmp(answer, "f") != 0);

clearConsole();

if (strcmp(answer, "file") == 0 || strcmp(answer, "f") == 0 ) {

printf("enter the file name: ");

scanf("%s", temp);

getchar();

file = fopen(temp, "r");

if (file == NULL) {

consoleError("no such file or directory ");

globalErrorFlag = 1;

} else {

while (fgets(temp, sizeof(temp), file) && !globalErrorFlag) {

if (strlen(text) + strlen(temp) < MAX\_LENGTH) {

strcat(text, temp);

} else {

fprintf(stderr, "exceeded limit of text size\n");

globalErrorFlag = 1;

}

}

fclose(file);

}

} else {

clearConsole();

printf("Enter text (end of text is empty line) and press ENTER:\n");

while (inputFlag && !globalErrorFlag) {

if (fgets(temp, MAX\_LENGTH, stdin) == NULL) {

consoleError("failed to input text");

globalErrorFlag = 1;

}

if (strcmp(temp, "\n") == 0 && !globalErrorFlag) {

inputFlag = 0;

}

if (strlen(text) + strlen(temp) >= MAX\_LENGTH && !globalErrorFlag && inputFlag) {

consoleError("exceeded limit of text size");

globalErrorFlag = 1;

}

if (!globalErrorFlag && inputFlag) {

strcat(text, temp);

}

}

}

if (!globalErrorFlag) {

clearConsole();

printf("Enter the word to replace (1 word without delimiters) and press ENTER:\n");

if (fgets(oldWord, MAX\_LENGTH, stdin) == NULL) {

consoleError("failed to input word");

} else {

removeNewline(oldWord);

if (isInvalid(oldWord)) {

consoleError("invalid word");

} else {

printf("Enter the word for replacement and press ENTER:\n");

if (fgets(newWord, MAX\_LENGTH, stdin) == NULL) {

consoleError("failed to input word");

} else {

removeNewline(newWord);

replaceWord(text, oldWord, newWord, result);

clearConsole();

printf("Original text:\n%s\n", text);

printf("\nProcessed text:\n%s\n", result);

}

}

}

}

return 0;

}

int isDelimiter(char c) {

return !(

c >= 'a' && c <= 'z' ||

c >= '0' && c <= '9' ||

c >= 'A' && c <= 'Z'

);

}

int isInvalid(const char \*str) {

return str[0] == '\0';

}

void consoleError(const char \*message) {

printf("Error: %s\n", message);

}

void removeNewline(char \*str) {

int i;

for (i = 0; i < MAX\_LENGTH; i++) {

if (str[i] == '\n' || str[i] == ' ') {

str[i] = '\0';

i = MAX\_LENGTH;

}

}

}

void replaceWord(char \*src, char \*oldWord, char \*newWord, char \*dest) {

char \*pos; /\* current position of oldWorld in src \*/

int len = 0, oldLen = strlen(oldWord); /\* len - length of destination string \*/

char \*firstSrc = src;

while ((pos = strstr(src, oldWord)) != NULL) {

/\* checking full word matching \*/

if ((pos == firstSrc || isDelimiter(\*(pos - 1))) && isDelimiter(\*(pos + oldLen))) { /\* it may be start of the string, so i added pos == src \*/

/\* i check current position in dest using dest + len and

add substring from src with pos - src length \*/

strncpy(dest + len, src, pos - src);

len += pos - src; /\* updating length \*/

strcpy(dest + len, newWord);

len += strlen(newWord);

src = pos + oldLen; /\* update src to position pos + oldLen \*/

} else {

strncpy(dest + len, src, pos - src + 1);

len += pos - src + 1;

src = pos + 1; /\* +1 bc i don't need infinite cycle \*/

}

}

strcpy(dest + len, src); /\* adding the rest of the src \*/

}

void clearConsole() {

#if defined(\_WIN32) || defined(\_WIN64)

system("cls");

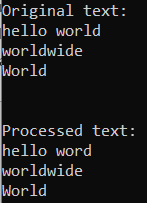
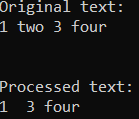
#else

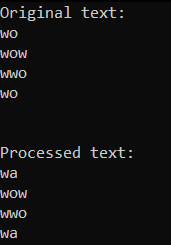
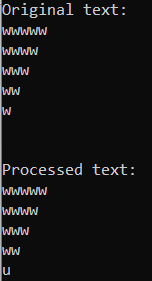
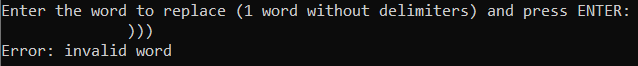
system("clear");

#endif

}

**Примеры выполнения программы**

****

****

**Заключение**

Заголовочные файлы:

<stdio.h>:

fclose & fopen (закрытие и открытие файла)

fgets (ввод строки)

printf (вывод в консоль)

getchar (очистка буфера)

scanf (ввод строки)

<string.h>

strcmp (лексикографическое сравнение строк)

strstr (получение вхождения в строке искомой подстроки)

strncpy (копирование части одной строки в другую)

strcpy (копирование одной строки в другую)

strcat (дозапись одной строки в другую)

strlen (получение длины строки)

<stdlib.h>

system (используется для очистки консоли)

Вывод: в результате выполнения курсовой работы были получены практические навыки работы с файлами и обработки строк в языке Си.