Санкт-Петербургский политехнический университет Петра Великого

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе № 5**

**Дисциплина:** Низкоуровневое программирование

**Тема:** Программирование на языке C

Выполнил студент гр. 3530901/90003 Зотов М. С.

(подпись)

Преподаватель Алексюк А.О.

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

**Оглавление**

[**1. ТЗ** 3](#_Toc69955171)

[**2. Метод решения** 3](#_Toc69955172)

[**3. Листинг программы** 4](#_Toc69955173)

[**3.1 main.c** 4](#_Toc69955174)

[**3.2 Replacement.h** 4](#_Toc69955175)

[**3.3 Replacement.c** 4](#_Toc69955176)

[**3.4 Error.h** 5](#_Toc69955177)

[**3.5 Error.c** 6](#_Toc69955178)

[**3.6 ConfigChecker.h** 6](#_Toc69955179)

[**3.7 ConfigChecker.c** 6](#_Toc69955180)

[**4. Коды ошибок** 7](#_Toc69955181)

[**5. Программа и методика испытаний.** 8](#_Toc69955182)

[**5.1 Ошибка 1: Неверное количество аргументов командной строки** 10](#_Toc69955183)

[**5.2 Ошибка 3: В файле конфигурации указано более одного слова для поиска** 10](#_Toc69955184)

[**5.3 Ошибка 4: В файле конфигурации указано более одного слова для замены** 10](#_Toc69955185)

[**5.4 Ошибка 5: В файле конфигурации слово для поиска не оканчивается на "** 11](#_Toc69955186)

[**5.5 Ошибка 6: В файле конфигурации слово для замены не оканчивается на "** 11](#_Toc69955187)

[**5.6 Ошибка 7: В файле конфигурации не найдено команды ни для замены, ни для поиска** 11](#_Toc69955188)

[**5.7 Ошибка 8: В файле конфигурации не найдено команды для замены** 12](#_Toc69955189)

[**5.8 Ошибка 9: В файле конфигурации не найдено команды для поиска** 12](#_Toc69955190)

[**5.9 Выполнение программы без ошибок** 12](#_Toc69955191)

[**5.9.1 Первый пример** 12](#_Toc69955192)

[**5.9.2 Второй пример** 13](#_Toc69955193)

[**5.9.3 Третий пример** 13](#_Toc69955194)

# **1. ТЗ**

Поиск и замена слов в тексте. В командной строке задаются имена файлов с исходным текстом, файлом конфигурации и файл с результатами. Пример файла конфигурации:

find = "один"

change = "два"

# **2. Метод решения**

Алгоритм решения:

1) Посимвольно читаем команды из *файла конфигурации*. Как только считанный символ совпал с нулевым символом у find = " или у change = ", начинаем смотреть, чтобы следующий считанный символ уже был равен первому символу у find = " или у change = " соответственно. Если равен, то смотрим, чтобы уже другой считанный символ был равен второму символу. Продолжаем так делать до тех пор, пока не дойдем до конца слова find = " или change = ". Как дошли, значит, дальше до символа "идут символы, составляющее слово на поиск (в случае find = "findWord") либо на замену (в случае change = "changeWord") Соответственно, надо на этом этапе создать счётчик j (который будет инкрементироваться при каждом новом символе), а также записывать считываемые символы в переменную findWord или changeWord до тех пор, пока не встретим символ ". Как дойдём до ", в j у нас будет храниться размер слова на поиск/замену. Сохраняем этот размер.

2) На данном этапе у нас есть слова на поиск/замену и их длины. **(!)** Начинаем так же посимвольно считывать текст из *входного файла* и сразу же записывать их в *выходной файл*, однако, как только считанный символ совпал с нулевым символом у findWord, во-первых, перестаем печатать считываемые символы в выходной файл – вместо этого записываем их в слово curWord, во-вторых, начинаем смотреть, чтобы следующий считанный символ уже соответствовал первому символу слова findWord. Если соответствует, то смотрим, чтобы уже другой считанный символ был равен второму символу findWord. Дальше ***3*** варианта:

А) Продолжаем так делать до тех пор, пока не дойдем до конца слова findWord (проверка дохода до конца делается через сравнение счетчика считанных символов в curWord и длины findWord). Как дошли, печатаем changeWord в *выходной файл* и возобновляем печать символов из *входного файла* сразу в *выходной*, то есть переходим в **(!)**

Б) Если не дошли до конца findWord, то есть встретили символ, не соответствующий findWord, то, ***если findWord[0] != считанному символу***, просто печатаем в *выходной файл* curWord и возобновляем печать символов из *входного файла* сразу в *выходной*, то есть переходим в **(!)**. ***Если же findWord[0] == считанному символу,*** то нужно просто изменить длину curWord на единицу и призвоить curWord[0] = считанный символ, переходить в **(!)** и печатать напрямую символы из входного файла в выходной **не нужно**.

# **3. Листинг программы**

# **3.1 main.c**

#include <windows.h>  
#include "Replacement.h"  
#include "ConfigChecker.h"  
  
int main(int argc, char \*args[]) {  
 SetConsoleOutputCP(**CP\_UTF8**);  
  
 int checkArgc = argcIsCorrect(argc);  
 if (checkArgc != 0)  
 return checkArgc;  
  
 char \*configFileName = args[1];  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int sizeFindWord;  
 int sizeChangeWord;  
 int checkConfig = getFindWordAndChangeWord(configFileName, findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 if (checkConfig != 0)  
 return checkConfig;  
  
 char \*inputFileName = args[2];  
 char \*outputFileName = args[3];  
 int returnCode = replaceWords(inputFileName, outputFileName, findWord, changeWord, sizeFindWord, sizeChangeWord);  
  
 return returnCode;  
}

# **3.2 Replacement.h**

#ifndef **LAB5\_REPLACEMENT\_H**#define **LAB5\_REPLACEMENT\_H**int argcIsCorrect(int argc);  
int replaceWords(char \*inputFileName, char \*outputFileName, char \*findWord, char \*changeWord, int sizeFindWord, int sizeChangeWord);  
  
#endif

# **3.3 Replacement.c**

#include <stdio.h>  
#include <stdlib.h>  
#include "Error.h"  
  
int argcIsCorrect(int argc) {  
 if (argc != 4)  
 return error(1, "Неверное количество аргументов командной строки. Ожидается 3 аргемента вида:\n"  
 "путь\\к\\конфигурации путь\\к\\входному\\тексту путь\\к\\выходному\\тексту");  
 else  
 return 0;  
}  
  
int replaceWords(char \*inputFileName, char \*outputFileName, char \*findWord, char \*changeWord, int sizeFindWord, int sizeChangeWord) {  
 FILE \*input, \*output;  
  
 input = fopen(inputFileName, "r");  
  
 if (input == **NULL**) {  
 int errorCode = error(10, "Файл с исходным текстом не удалось прочитать. Имя введенного входного файла:\n");  
 printf("%s", inputFileName);  
 return errorCode;  
 }  
  
 output = fopen(outputFileName, "w");  
  
 if (output == **NULL**) {  
 fclose(input);  
 int errorCode = error(11, "Произошла ошибка при попытке работы с файлом, содержащим результат. Имя введенного выходного файла:\n");  
 printf("%s", outputFileName);  
 return errorCode;  
 }  
  
 char \*curWord = (char \*)malloc(0);  
  
 if (curWord == **NULL**) {  
 return error(16, "Ошибка выделения памяти под curWord");  
 }  
  
 int i = 0;  
 char ch;  
 while ((ch = fgetc(input)) != **EOF**) {  
 if (ch == findWord[i]) {  
 curWord = realloc(curWord, (i + 1) \* sizeof(char));  
  
 if (curWord == **NULL**)   
 return error(17, "Ошибка перераспределения памяти под curWord");  
  
 curWord[i] = ch;  
 if (i == sizeFindWord - 1) {  
 for (int j = 0; j < sizeChangeWord; j++)  
 fputc(changeWord[j], output);  
 i = 0;  
 } else i++;  
 } else if (i != 0) {  
 for (int j = 0; j < i; j++)  
 fputc(curWord[j], output);  
 if (ch != findWord[0]) {  
 fputc(ch, output);  
 i = 0;  
 } else {  
 curWord = realloc(curWord, sizeof(char));  
  
 if (curWord == **NULL**)   
 return error(17, "Ошибка перераспределения памяти под curWord");  
  
 curWord[0] = ch;  
 i = 1;  
 }  
 } else {  
 fputc(ch, output);  
 i = 0;  
 }  
 }  
  
 fclose(input);  
 fclose(output);  
  
 free(curWord);  
  
 printf("Слова были успешно заменены");  
  
 return 0;  
}

# **3.4 Error.h**

#ifndef **LAB5\_ERROR\_H**#define **LAB5\_ERROR\_H**int error(int errorCode, char \*msg);  
  
#endif

# **3.5 Error.c**

#include <stdio.h>  
  
int error(int errorCode, char \*msg) {  
 printf(“Ошибка %d: %s”, errorCode, msg);  
 return errorCode;  
}

# **3.6 ConfigChecker.h**

#ifndef **LAB5\_CONFIGCHECKER\_H**#define **LAB5\_CONFIGCHECKER\_H**int getFindWordAndChangeWord(char \*configFileName, char \*findWord, char \*changeWord, int \*sizeFindWord, int \*sizeChangeWord);  
  
#endif

# **3.7 ConfigChecker.c**

#include <stdio.h>  
#include <stdlib.h>  
#include <stdbool.h>  
#include "Error.h"  
  
int showExpectedConfig(int errorCode, char \*msg) {  
 error(errorCode, msg);  
 printf("\nОжидаемый вид файла конфигурации:\nfind = \"findWord\"\nchange = \"changeWord\"");  
 return errorCode;  
}  
  
int getFindWordAndChangeWord(char \*configFileName, char \*findWord, char \*changeWord, int \*sizeFindWord, int \*sizeChangeWord) {  
 FILE \*config = fopen(configFileName, "r");  
  
 if (config == **NULL**) {  
 int errorCode = error(2, "Файл конфигурации не удалось прочитать. Имя введенного файла конфигурации:\n");  
 printf("%s", configFileName);  
 return errorCode;  
 }  
  
 char find[] = "find = \"";  
 char change[] = "change = \"";  
  
 int sizeFind = sizeof (find) / sizeof (char) - 1;  
 int sizeChange = sizeof (change) / sizeof (char) - 1;  
  
 \*sizeFindWord = -1;  
 \*sizeChangeWord = -1;  
  
 int i = 0;  
 char ch;  
  
 while ((ch = fgetc(config)) != **EOF**) {  
 if (ch == find[i] || ch == change[i]) {  
 **bool** findWordIsBeingRead = (i == sizeFind - 1 && ch == find[i]);  
 **bool** changeWordIsBeingRead = (i == sizeChange - 1 && ch == change[i]);  
 if (findWordIsBeingRead || changeWordIsBeingRead) {  
 if (\*sizeChangeWord != -1 && \*sizeFindWord != -1) {  
 fclose(config);  
 if (findWordIsBeingRead)  
 return showExpectedConfig(3, "В файле конфигурации указано более одного слова для поиска");  
 else  
 return showExpectedConfig(4, "В файле конфигурации указано более одного слова для замены");  
 }  
  
 int j = 0;  
 while ((ch = fgetc(config)) != '"') {  
 if (ch == **EOF**) {  
 fclose(config);  
 if (findWordIsBeingRead)  
 return showExpectedConfig(5, "В файле конфигурации слово для поиска не оканчивается на \"");  
 else  
 return showExpectedConfig(6, "В файле конфигурации слово для замены не оканчивается на \"");  
 }  
 if (findWordIsBeingRead) {  
 findWord = realloc(findWord, (j + 1) \* sizeof(char));  
  
 if (findWord == **NULL**) {  
 fclose(config);  
 return error(14, "Ошибка перераспределения памяти под findWord");  
 }  
  
 findWord[j] = ch;  
 }  
 else {  
 changeWord = realloc(changeWord, (j + 1) \* sizeof(char));  
  
 if (changeWord == **NULL**) {  
 fclose(config);  
 return error(15, "Ошибка перераспределения памяти под changeWord");  
 }  
  
 changeWord[j] = ch;  
 }  
 j++;  
 }  
 if (findWordIsBeingRead) \*sizeFindWord = j;  
 else \*sizeChangeWord = j;  
 } else i++;  
 } else i = 0;  
 }  
  
 fclose(config);  
  
 **bool** changeWordNotFound = \*sizeChangeWord == -1;  
 **bool** findWordNotFound = \*sizeFindWord == -1;  
 if (changeWordNotFound || findWordNotFound) {  
 if (changeWordNotFound && findWordNotFound)  
 return showExpectedConfig(7, "В файле конфигурации не найдено команды ни для замены, ни для поиска");  
 else if (changeWordNotFound)  
 return showExpectedConfig(8, "В файле конфигурации не найдено команды для замены");  
 else  
 return showExpectedConfig(9, "В файле конфигурации не найдено команды для поиска");  
 }  
  
 return 0;  
}

# **4. Коды ошибок**

Ошибка 1: Неверное количество аргументов командной строки

Ошибка 2: Файл конфигурации не удалось прочитать

Ошибка 3: В файле конфигурации указано более одного слова для поиска

Ошибка 4: В файле конфигурации указано более одного слова для замены

Ошибка 5: В файле конфигурации слово для поиска не оканчивается на "

Ошибка 6: В файле конфигурации слово для замены не оканчивается на "

Ошибка 7: В файле конфигурации не найдено команды ни для замены, ни для поиска

Ошибка 8: В файле конфигурации не найдено команды для замены

Ошибка 9: В файле конфигурации не найдено команды для поиска

Ошибка 10: Файл с исходным текстом не удалось прочитать

Ошибка 11: Произошла ошибка при попытке работы с файлом, содержащим результат

Ошибка 12: Ошибка выделения памяти под findWord

Ошибка 13: Ошибка выделения памяти под changeWord

Ошибка 14: Ошибка перераспределения памяти под findWord

Ошибка 15: Ошибка перераспределения памяти под changeWord

Ошибка 16: Ошибка выделения памяти под curWord

Ошибка 17: Ошибка перераспределения памяти под curWord

# **5. Программа и методика испытаний.**

Как можно видеть по пункту **3.1**, в функции main никаких проверок на ошибки нет – они есть в argcIsCorrect(), getFindWordAndChangeWord() и replaceWords(). В этой связи мне показалось удобным написать следующие файлы для проверки того, как возникающие ошибки обрабатываются:

**test.c:**

#include <stdlib.h>  
#include "../Replacement.h"  
#include "../ConfigChecker.h"  
  
int sizeFindWord;  
int sizeChangeWord;  
  
int testCorrect1() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 char \*config\_correct\_1 = "..\\texts\\correct\\config\_correct\_1.txt";  
 int checkConfig = getFindWordAndCheckWord(config\_correct\_1, findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 char \*input\_correct\_1 = "..\\texts\\correct\\input\_correct\_1.txt";  
 char \*output\_correct\_1 = "..\\texts\\correct\\results\\output\_correct\_1.txt";  
 int returnCode = replaceWords(input\_correct\_1, output\_correct\_1, findWord, changeWord, sizeFindWord, sizeChangeWord);  
 return 0;  
}  
int testCorrect2() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 char \*config\_correct\_2 = "..\\texts\\correct\\config\_correct\_2.txt";  
 int checkConfig = getFindWordAndCheckWord(config\_correct\_2, findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 char \*input\_correct\_2 = "..\\texts\\correct\\input\_correct\_2.txt";  
 char \*output\_correct\_2 = "..\\texts\\correct\\results\\output\_correct\_2.txt";  
 int returnCode = replaceWords(input\_correct\_2, output\_correct\_2, findWord, changeWord, sizeFindWord, sizeChangeWord);  
 return returnCode;  
}  
int testCorrect3() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 char \*config\_correct\_3 = "..\\texts\\correct\\config\_correct\_3.txt";  
 int checkConfig = getFindWordAndCheckWord(config\_correct\_3, findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 char \*input\_correct\_3 = "..\\texts\\correct\\input\_correct\_3.txt";  
 char \*output\_correct\_3 = "..\\texts\\correct\\results\\output\_correct\_3.txt";  
 int returnCode = replaceWords(input\_correct\_3, output\_correct\_3, findWord, changeWord, sizeFindWord, sizeChangeWord);  
 return returnCode;  
}  
  
  
int testIncorrect2() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int returnCode = getFindWordAndCheckWord("..\\texts\\incorrect\\config\_incorrect\_2.txt", findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 return returnCode;  
}  
int testIncorrect3() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int returnCode = getFindWordAndCheckWord("..\\texts\\incorrect\\config\_incorrect\_3.txt", findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 return returnCode;  
}  
int testIncorrect4() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int returnCode = getFindWordAndCheckWord("..\\texts\\incorrect\\config\_incorrect\_4.txt", findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 return returnCode;  
}  
int testIncorrect5() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int returnCode = getFindWordAndCheckWord("..\\texts\\incorrect\\config\_incorrect\_5.txt", findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 return returnCode;  
}  
int testIncorrect6() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int returnCode = getFindWordAndCheckWord("..\\texts\\incorrect\\config\_incorrect\_6.txt", findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 return returnCode;  
}  
int testIncorrect7() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int returnCode = getFindWordAndCheckWord("..\\texts\\incorrect\\config\_incorrect\_7.txt", findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 return returnCode;  
}  
int testIncorrect8() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int returnCode = getFindWordAndCheckWord("..\\texts\\incorrect\\config\_incorrect\_8.txt", findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 return returnCode;  
}  
int testIncorrect9() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int returnCode = getFindWordAndCheckWord("..\\texts\\incorrect\\config\_incorrect\_9.txt", findWord, changeWord, &sizeFindWord, &sizeChangeWord);  
 return returnCode;  
}  
int testIncorrect10() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int returnCode = replaceWords("..\\texts\\correct\\config\_correct\_1.txt", "", findWord, changeWord, sizeFindWord, sizeChangeWord);  
 return returnCode;  
}  
int testIncorrect11() {  
 char \*findWord = (char \*)malloc(0);  
 char \*changeWord = (char \*)malloc(0);  
 int returnCode = replaceWords("..\\texts\\correct\\config\_correct\_1.txt","", findWord, changeWord, sizeFindWord, sizeChangeWord);  
 return returnCode;  
}  
  
int testArgc(int argc) { return !argcIsCorrect(argc) ? 0 : 1; }  
  
int doTests() {  
 return 0;  
}

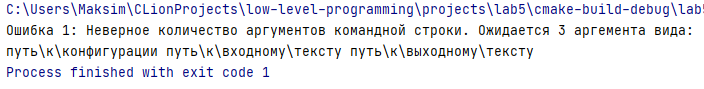
**main.c:**

#include <windows.h>  
#include "tests/test.h"  
  
int main(int argc, char \*args[]) {  
 SetConsoleOutputCP(**CP\_UTF8**);  
  
 return doTests();  
}

# **5.1 Ошибка 1: Неверное количество аргументов командной строки**

Пропишем в doTests() из **test.c** следующее: return testArgc(5);.

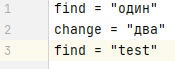
Вывод в консоль:



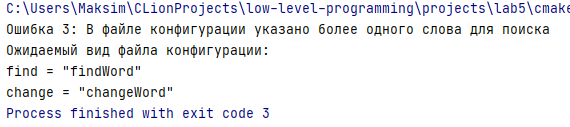
Далее я не буду комментировать, какой код для какого случая прописывается в doTests() – вместо этого просто будет показано, обработка каких файлов к какой ошибке привела.

# **5.2 Ошибка 3: В файле конфигурации указано более одного слова для поиска**

Конфигурация:

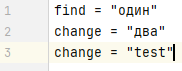


Вывод:

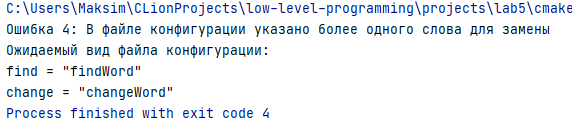


# **5.3 Ошибка 4: В файле конфигурации указано более одного слова для замены**

Конфигурация:



Вывод:

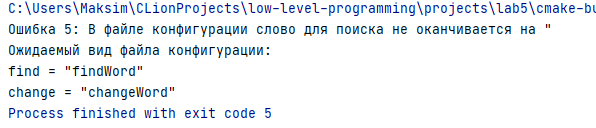


# **5.4 Ошибка 5: В файле конфигурации слово для поиска не оканчивается на "**

Конфигурация:



Вывод:

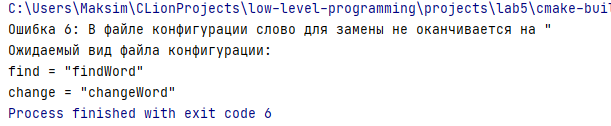


# **5.5 Ошибка 6: В файле конфигурации слово для замены не оканчивается на "**

Конфигурация:



Вывод:

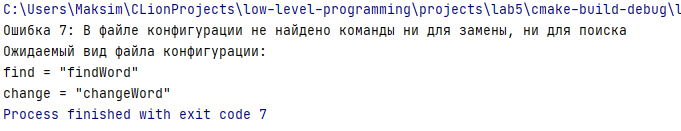


# **5.6 Ошибка 7: В файле конфигурации не найдено команды ни для замены, ни для поиска**

Конфигурация:



Вывод:

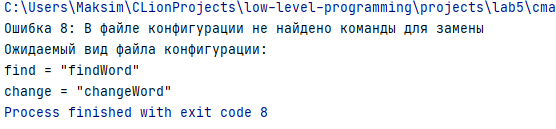


# **5.7 Ошибка 8: В файле конфигурации не найдено команды для замены**

Конфигурация:



Вывод:

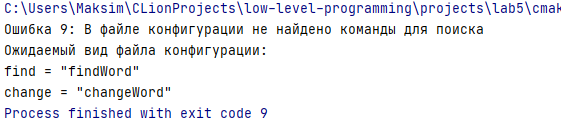


# **5.8 Ошибка 9: В файле конфигурации не найдено команды для поиска**

Конфигурация:



Вывод:



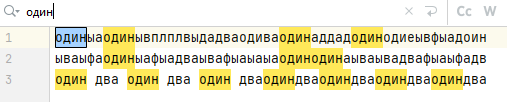
# **5.9 Выполнение программы без ошибок**

# **5.9.1 Первый пример**

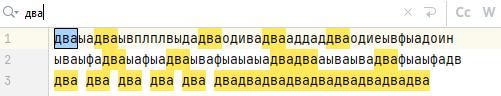
Конфигурация:



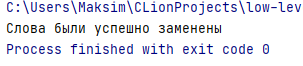
Входной файл:



Выходной файл:



Вывод:

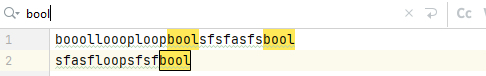


# **5.9.2 Второй пример**

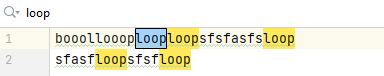
Конфигурация:



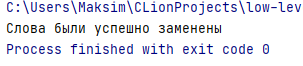
Входной файл:



Выходной файл:



Вывод:



# **5.9.3 Третий пример**

Конфигурация:



Входной файл:



Выходной файл:



Вывод:

