Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Волгоградский государственный технический университет»

Факультет

\_\_\_\_\_\_\_\_\_\_\_\_электроники и вычислительной техники\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра

\_\_\_\_\_\_\_\_\_программного обеспечения автоматизированных систем\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  Заведующий кафедрой ПОАС  \_\_\_\_\_\_\_\_\_ \_\_\_Орлова Ю.А.\_\_\_\_\_  (подпись) (расшифровка подписи)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_ г. |

ЗАДАНИЕ

на практику по получению первичных профессиональных умений и навыков

(2-й курс, 4-й семестр)

Студенту\_ Степанову С.В. Группа\_\_\_\_ПрИн-266\_\_\_\_\_\_

(фамилия, имя, отчество)

1. Изучить \_ГОСТ 19, основные проблемы и правила обратной транслитерации, стандартную библиотеку C++ рекурсивные алгоритмы

2. Разработать\_\_\_ рабочую документацию и программу, на основе разработанной ранее спецификации\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Произвести\_тестирование и контроль качества разработанного кода\_\_\_\_\_\_

|  |  |
| --- | --- |
| Дата выдачи задания  Руководитель практики университета | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 201\_ г.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Волгоградский государственный технический университет»

Факультет

\_\_\_\_\_\_\_\_\_\_\_\_электроники и вычислительной техники\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра

\_\_\_\_\_\_\_\_\_программного обеспечения автоматизированных систем\_\_\_\_\_\_\_\_\_

ОТЧЕТ

о практике по получению первичных профессиональных умений и навыков

(2-й курс, 4-й семестр)

на кафедре «Программное обеспечение автоматизированных систем»

|  |  |
| --- | --- |
| Руководитель практики от  университета | \_\_\_доцент\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сычев О.А.  должность подпись |

Студенту гр. ПрИн-266 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Степанову С.В.

подпись

Отчёт защищен с оценкой \_\_\_\_\_\_\_\_\_

Волгоград 2019 г.

Содержание

[1. Рабочая документация 4](file:///D:\ucheba\kinpo\документация%20к%20калькулятору\отчёт.doc#_Toc517868584)

[1.1 Вызов программы 4](file:///D:\ucheba\kinpo\документация%20к%20калькулятору\отчёт.doc#_Toc517868585)

[1.2 Сообщения об ошибках 4](file:///D:\ucheba\kinpo\документация%20к%20калькулятору\отчёт.doc#_Toc517868586)

[1.3 Код ключевых функций 6](file:///D:\ucheba\kinpo\документация%20к%20калькулятору\отчёт.doc#_Toc517868587)

# 1. Рабочая документация

## 1.1 Вызов программы

Программа работает в режиме аргументов командной строки. В режиме аргументов командной строки программа принимает на вход через пробел:

1. Путь к самой программе;

2) TXT – файл, содержащий текст для обратной транслитерации;

3) TXT – файл, используемый для вывода готового транслитированного текста на кириллице;

4) TXT – файл, содержащий словарь (набор правил) для транслитерации (формат файла представлен в приложении Б технического задания).

Если командная строка запущена не в папке с программой, то требуется указать полный путь всех файлов.

Пример вызова программы из командной строки:

…\ myTranslit.exe infile.txt outfile.txt abc.txt

## 1.2 Сообщения об ошибках

Программа выдает пользователю сообщения об ошибках.

В случае получения сообщения “Неверно заданы параметры Формат myTranslit.exe infile.txt outfile.txt abc.txt” пользователю стоит проверить правильность ввода аргументов командной строки.

В случае получения сообщения “Не удалось открыть входной файл” пользователю стоит проверить наличие файла infile.txt по указанному пользователем пути в командной строке.

В случае получения сообщения “Не удалось открыть файл со словарем” пользователю стоит проверить наличие файла abc.txt по указанному пользователем пути в командной строке.

В случае получения сообщения “Не удалось открыть выходной файл” пользователю стоит проверить наличие файла outfile.txt по указанному пользователем пути в командной строке.

В случае получения сообщения “Есть латинские представления кириллицы с длиной больше 3 символов на позиции: [номер позиции]” пользователю стоит проверить в файле abc.txt по указанной позиции латинское представление символа (размер латинского представления не должен превышать 3-х символов).

В случае получения сообщения “!В словаре есть одинаковые латинские представления букв!” пользователю стоит проверить в файле abc.txt наличие повторяющихся представлений в латинской части словаря.

В случае получения сообщения "!В словаре есть одинаковые кириллические представления букв!" пользователю стоит проверить в файле abc.txt наличие повторяющихся представлений в кириллической части словаря.

В случае получения сообщения “!Не совпадает кол-во кириллических и латинских представлений в словаре! Кириллических: [кол-во кир-их] Латинских: [кол-во лат-их]” пользователю стоит проверить кол-во латинских и кириллических представлений символов в файле abc.txt (кол-во латинских и кириллических представлений должно совпадать)

В случае получения сообщения "Не латинское представление в словаре на позиции: [номер позиции]" пользователю стоит наличие нелатинских символов в латинской части представления словаря в файле abc.txt.

## 

## 1.3 Код ключевых функций

bool abcLatinValidator(std::vector<std::string>& lat)

{

std::vector<int> not\_lat\_pos;

for(size\_t i=0; i<lat.size(); i++)

{

for(size\_t j=0; j<lat[i].size(); j++)

{

if(!isalpha(lat[i][j]))

{

not\_lat\_pos.push\_back(i+1);

break;

}

}

}

if(!not\_lat\_pos.empty())

{

std::cerr << "Не латинское представление в словаре на позиции: ";

for(std::vector<int>::iterator it = not\_lat\_pos.begin(); it!=not\_lat\_pos.end(); ++it)

{

std::cerr << \*it << " ";

}

std::cerr << std::endl;

return false;

}

else

{

return true;

}

}

bool abcSizesValidator(std::vector<std::string>& lat)

{

for(size\_t i=0; i<lat.size(); i++)

{

if(lat[i].size()>3)

{

std::cerr <<"Есть латинские представления кириллицы с длиной больше 3 символов на позиции: " << i+1 << std::endl;

return false;

}

}

return true;

}

bool abcOverlapValidator(std::vector<std::string>& lat,std::vector<std::string>& rus)

{

for(size\_t i=0; i<lat.size()-1; i++)

{

for(size\_t j=i+1; j<lat.size(); j++)

{

if(lat[i]==lat[j])

{

std::cerr<< "!В словаре есть одинаковые латинские представления букв!" <<std::endl;

return false;

}

}

}

for(size\_t i=0; i<rus.size()-1; i++)

{

for(size\_t j=i+1; j<rus.size(); j++)

{

if(rus[i]==rus[j])

{

std::cerr<< "!В словаре есть одинаковые кириллические представления букв!" <<std::endl;

return false;

}

}

}

return true;

}

bool abcCountFileValidator(std::string& rus, std::string& lat)

{

int lat\_count\_val = 0;

int rus\_count\_val = 0;

for(size\_t i = 0; i < lat.size(); i++)

{

if(lat[i]==';' && lat[i]!=lat[i+1])

lat\_count\_val++;

}

for(size\_t i = 0; i < rus.size(); i++)

{

if(rus[i]==';' && rus[i]!=rus[i+1])

rus\_count\_val++;

}

if(rus\_count\_val!=lat\_count\_val)

{

std::cerr << "!Не совпадает кол-во кириллических и латинских представлений в словаре!\nКириллических: " << rus\_count\_val+1 << "\nЛатинских: "<< lat\_count\_val+1 << std::endl;

return false;

}

else

{

return true;

}

}

int canTranslit(const std::string& src, const std::map<std::string,std::string>& abc, int translitPos)

{

std::string src\_buf;

for(int i = translitPos; i<src.size(); i++)

{

src\_buf.push\_back(src[i]);

if(src.size()-translitPos==1 && abc.find(src\_buf)!=abc.end())

{

src\_buf.clear();

return 1;

}

else if(src.size()-translitPos>=1 && abc.find(src\_buf)!=abc.end() && canTranslit(src,abc,i+1))

{

src\_buf.clear();

return 1;

}

else if(src.size()-translitPos>=2 && abc.find(src\_buf+ src[i+1])!=abc.end() && canTranslit(src,abc,i+2))

{

src\_buf.clear();

return 2;

}

else if(src.size()-translitPos>=3 && abc.find(src\_buf+ src[i+1] + src[i+2])!=abc.end() && canTranslit(src,abc,i+3))

{

src\_buf.clear();

return 3;

}

else if(!isalpha(src[translitPos]))

{

src\_buf.clear();

return 4;

}

else

{

src\_buf.clear();

return 0;

}

}

}

std::string Translit(const std::string& src, const std::map<std::string,std::string> abc)

{

std::string res;

std::string src\_buf;

int can\_translit = 0;

for(int i=0; i<src.size(); i++)

{

if(isalpha(src[i]))

{

can\_translit = canTranslit(src,abc,i);

if(can\_translit==1)

{

src\_buf.push\_back(src[i]);

res.append(abc.find(src\_buf)->second);

}

else if(can\_translit==2)

{

src\_buf.push\_back(src[i]);

src\_buf.push\_back(src[i+1]);

res.append(abc.find(src\_buf)->second);

i++;

}

else if(can\_translit==3)

{

src\_buf.push\_back(src[i]);

src\_buf.push\_back(src[i+1]);

src\_buf.push\_back(src[i+2]);

res.append(abc.find(src\_buf)->second);

i+=2;

}

else

{

while(isalpha(src[i]))

{

res.append("~");

i++;

}

i--;

}

}

else

{

res.push\_back(src[i]);

}

src\_buf.clear();

}

return res;

}