Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Волгоградский государственный технический университет»

УТВЕРЖДАЮ:  
Зав. Кафедрой ПОАС  
\_\_\_\_\_\_\_доц., д.т.н. Орлова Ю. А.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

Качество и надежность программного обеспечения

«Обратная транслитерация»

Описание программы

СОГЛАСОВАНО: РАЗРАБОТЧИК:

Руководитель работы Студент группы ПрИн-266  
доцент кафедры ПОАС

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сычев О.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Степанов С.В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

НОРМОКОНТРОЛЕР:

преподаватель кафедры ПОАС

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Матюшечкин Д.С.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

2019 г.

Содержание

1 Общие сведения…………………………………………………………………3

2 Функциональное назначение…………………………………………………...3

3 Описание логических структур………………………………………………...3

4 Используемые технические средства……………………………...…………..4

5 Вызов и загрузка……………………………………………...…………………4

6 Выходные и входные данные…………………………………...……………...4

Приложение А. Описание структур данных…………………………...………..5

Приложение Б. Описание алгоритмов функций…………………………...……6

Приложение В. Диаграмма вызовов функций…………………………...…….13

Приложение Г. Диаграмма потоков данных……………………………………14

Приложение Д. Примеры транслитерации…………………...………………...15

1 Общие сведения

Название программы – “Reverse Transliteration”

Для корректного функционирования программы необходима операционная система Windows 7 и выше.

Программа написана на языке C++ с использованием его стандартных библиотек.

2 Функциональное назначение

Данная программа автоматически производит обратную транслитерацию из латиницы в кириллицу. При возникновении ситуации с невозможной транслитерации слово заменяется на символы “~” (Пример в приложении Д)

3 Описание логической структуры

Работа программы логически разделена на несколько частей:

- считывание входных данных;

- обратная транслитерация;

- вывод результата

Описание используемых в программе структур, данных приведено в приложении А, алгоритмы основных функций – в приложении Б, диаграмма вызова функций – в приложении B, диаграмма потоков данных – в приложении Г.

4 Используемые технические средства

Описание требуемых технических средств содержится в Техническом задании в пункте 3.3.

5 Вызов и загрузка

Программа запускается посредством командной строки с указанием 3 параметров – путь к файлу с текстом для транслитерации, путь к файлу для записи транслитированного текста, путь к файлу со словарем - алфавитом.

Пример командной строки: myTranslit.exe infile.txt outfile.txt abc.txt.

6 Входные и выходные данные

Формат входных и выходных данных описан в пункте 3.4 технического задания.

Приложение А

Описание используемых в программе структур данных

std::vector<char> Buf - буфер для записи считанного текста для транслитерации;

std::vector<std::string> rus - массив для хранения кириллической части словаря (будут использованы как значения);

std::vector<std::string> lat - массив для хранения латинской части словаря (будут использованы как ключи);

std::string buff[2] - массив строк для считывания словаря, buff[0] – содержит кириллический алфавит в виде строки, buff[1] – содержит латинское представление кириллического алфавита в виде строки;

std::map<std::string,std::string> alphabet - словарь хранящий считанный словарь - алфавит из файла, где ключ - латинское представление, а значение – кириллическое;

Приложение Б

Описание алгоритмов функций

Главная функция программы:

int main(int argc, char \*\*argv)

Входные данные:

argc – количество переданных аргументов командной строки

argv[] – аргументы командной строки

argv[1] – аргумент файла с транслитированным текстом

argv[2] – аргумент файла для вывода текста на кириллице

argv[3] – аргумент файла со словарем - алфавитом

Выходные данные:

Обратно транслитированный текст на кириллице, записанный в файл outfile.txt.

Алгоритм функции:

1. Открыть файлы для считывания и записи
2. Считать транслитированный текст в буферную строку
3. Проверить словарь- алфавит на наличие ошибок
4. Записать словарь-алфавит в словарь
5. Транслитировать текст на латинице из буферной строки
6. Записать результат обратной транслитерации в файл ответа

Функция обратной транслитерации:

/\*!

\* Функция проверяет корректность считанных данных

\*\param [in] src - строка с транслитируемым текстом

\*\param [in] abc - словарь с правилами обратной трансилтерации текста

\*\return - транслитированный текст на кириллице

\*/

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

Алгоритм функции:

1.Для каждого символа текста…

1.1. Если символ латинский…

1.1.1. Если можно транслитировать символ…

1.1.1.1. Скопировать символ в буферную строку

1.1.1.2. Транслитировать и добавить к результирующей строке

1.1.2. Иначе Если можно транслитировать 2 символа…

1.1.2.1. Скопировать 2 символа в буферную строку

1.1.2.2. Транслитировать и добавить к результирующей строке

1.1.3. Иначе Если можно транслитировать 3 символа…

1.1.3.1. Скопировать 23символа в буферную строку

1.1.3.2. Транслитировать и добавить к результирующей строке

1.1.4. Иначе…

1.1.4.1. Заменить нетранслитируемую комбинацию символами ‘~’

1.2.Иначе…

1.2.1. Добавить текущий символ к результирующей строке

1.3. Очистить буферную строку

2.Вернуть значение результирующей строки

Функция проверки возможности транслитерации текущей последовательности.

/\*Функиция проверяет, можно ли транслитировать текущую последовательность символов, не заводя транслитерацию в тупик

\*\param [in] src - транлитируемый текст

\*\param [in] abc - словарь правил транслитерации

\*\param [in] translitPos - позиция транслитерации,

\*\param [in] translitSize - размер последовательности транслитерации

\*/

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

Алгоритм функции:

1.Для каждого символа, начиная с позиции транслитерации…

1.1. Если символ один или последний и его можно траслитировать…

1.1.1. Сигнал о возможности транслитерации символа

1.2. Иначе Если можно траслитировать текущий символ и следующий

1.2.1. Сигнал о возможности транслитерации символа

1.3. Иначе Если можно траслитировать текущие 2 символа и следующий

1.3.1. Сигнал о возможности транслитерации 2 символов

1.4. Иначе Если можно траслитировать текущие 3 символа и следующий

1.4.1. Сигнал о возможности транслитерации 3 символов

1.5. Иначе Если на позиции транслитерации не латинская буква

1.5.1. Сигнал о игнорировании символа

1.6 Иначе…

1.6.1 Сигнал о невозможности транслитерации

Функция проверки кол-во представлений символов в файле-словаре.

/\*Функиция проверяет кол-во кириллических и латинских представлений символов

\*\param [in] rus – кириллическая часть алфавита

\*\param [in] lat – латинская часть алфавита

\*/

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

Алгоритм функции:

1. Для каждого символа в строке с кириллическим представлением

1.1. Если найден символ ‘;’…

1.1.1. Посчитать его

2. Для каждого символа в строке с латинским представлением

2.1. Если найден символ ‘;’…

2.1.1. Посчитать его

3. Если кол-во представлений не совпадает…

3.1. Вывести сообщение об ошибке

3.2. Сигнал об ошибке

4. Иначе…

4.1. Сигнал о продолжении выполнении программы

Функция проверки повторения представлений в словаре.

/\*Функиция проверяет повторения представлений в словаре

\*\param [in] lat - кириллическая часть алфавита

\*\param [in] tus - латинская часть алфавита

\*/

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

Алгоритм функции:

1. Для каждого латинского представления…

1.1. Для каждого последующего латинского представления…

1.1.1. Если представления одинаковые…

1.1.1.1. Вывести сообщение об ошибке

1.1.1.2. Сигнал об ошибке

2. Для каждого кириллического представления…

2.1. Для каждого последующего кириллического представления…

2.1.1. Если представления одинаковые…

2.1.1.1. Вывести сообщение об ошибке

2.1.1.2. Сигнал об ошибке

3.Сигнал о продолжении выполнении программы

Функция проверки максимального размера латинского представления.

/\*Функиция проверяет максимальный размер представления в словаре

\*\param [in] lat - латинская часть алфавита

\*/

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

Алгоритм функции:

1. Для каждого латинского представления символа
   1. Если размер латинского представления больше 3…
      1. Вывести сообщение об ошибке
      2. Сигнал об ошибке
2. Сигнал о продолжении выполнении программы

Функция проверки содержания нелатинских символов в латинских представлениях кириллицы.

/\*Функиция проверяет есть ли нелатинские символы в части с латинским представлением алфавита

\*\param [in] lat - латинская часть алфавита

\*/

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

Алгоритм функции:

1. Для каждого представления…

1.1. Для каждого символа представления…

1.1.1. Если не латинский символ, то…

1.1.1.1. Запомнить позицию нелатинского представления

2.Если найдено нелатинское представление

2.1. Вывести сообщение об ошибке

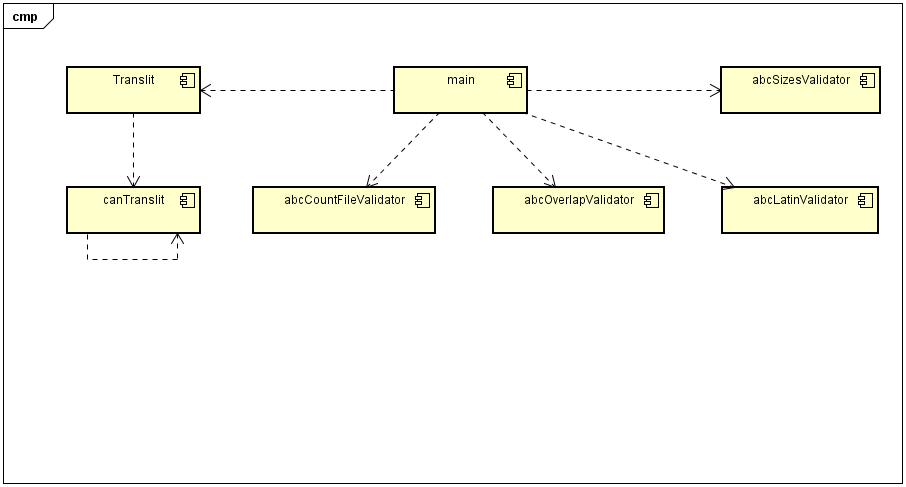
2.2. Сигнал о завершении программы

3. Иначе…

3.1. Сигнал о продолжении выполнения программы

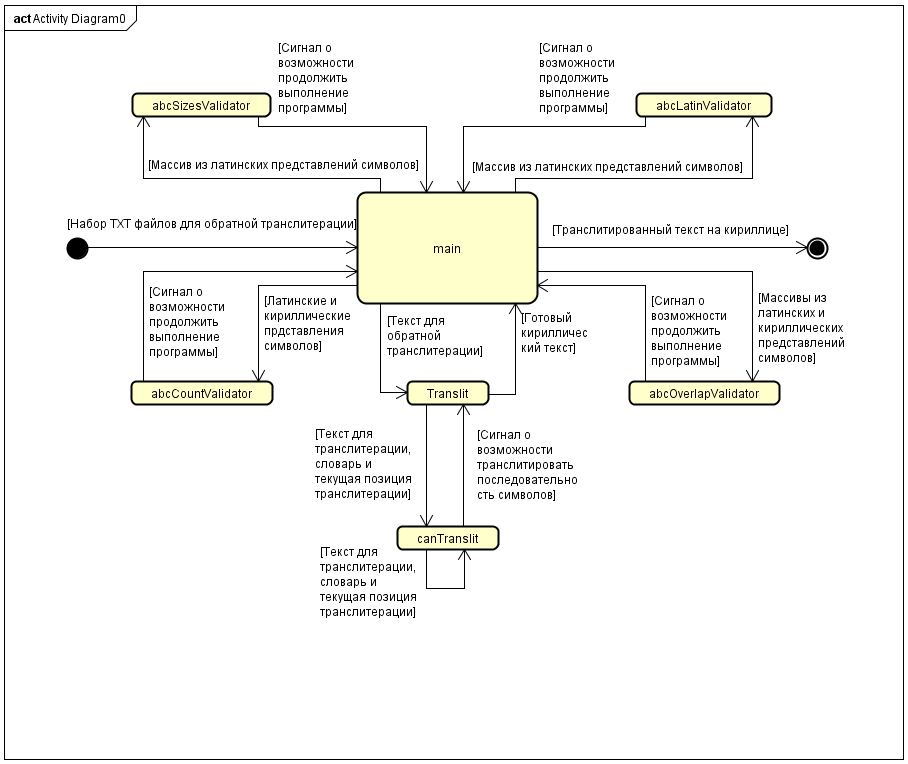
Приложение В

Диаграмма вызова функций



Приложение Г

Диаграмма потоков данных



Приложение Д

Примеры транслитерации.

|  |  |  |
| --- | --- | --- |
| Словарь | Текст для транслитерации | Текст на выходе |
| а;б;в  a;b;v | abv, abg, abb,zhm | абв, ~~~, абб, ~~~ |
| а;б;в  a;ab;bb | abbb, abb, ab, a, b | бв, ав, б, а, ~ |
| ч;е;с;н;о;к  ch;e;s;n;o;k | chesnok | чеснок |