Tarnowskie Góry, 26.12.17

Semester: 1

Group: 1

Section: 3

Computer Programming Laboratory

Author: Marek Paśnik

E-mail: marepas035@student.polsl.pl

Tutor: Piotr Fabian

1. Task topic

Translator. Write a program that “translates” a text file by replacing words with corresponding words read from a separate dictionary. The dictionary should be read from a file once and later kept in memory.

1. Project analysis

The program needs several types of data structures to keep necessary information during its work, with loading them once. These are:

* input file given by user
* dictionary file of wanted language, properly formatted to use

The program on its start checks whether both input file and dictionary file exists. If both were positive, the files are opened and words from input file are compared with loaded dictionary, if the word is found in dictionary it will be replaced with its translated version and printed to the output file. If the given word isn’t in the dictionary, program will omit the translation and print original (non-translated) word to the output file. These instructions are executed for each word in the input file.

1. External specification

This program is very easy to use for the user, with built-in help. As it is a command line program you need to use command line prompt to get it work. It can be run by typing:

Translator.exe –d path-for-dictionary-file -i path-for-input-file

For example:

Translator.exe –d PolishEnglishDictionary.txt –i myOriginalFile.txt

For arguments presented above the program will translate the input file given in Polish to English and create output file with original name file and added \_EN postfix.

When input path is given incorrectly (file not exists, etc.) the program terminates with information:

- for wrong dictionary file



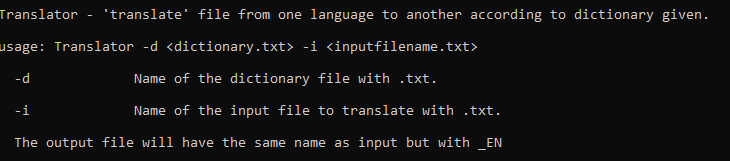
- for wrong input file.



- if no parameters passed.



Program help:



1. Internal specification

The program are contains two classes:

Class Translator which contains:

* Methods:

void loadDict(std::string dictFileName) - loads dictionary from given file name and assigns to the pointer

std::string translateWord(std::string word) - translating one word using loaded dictionary and returns it, (if the word is not found in dictionary it will return originally given word)

void translateFile(std::string fileToTranslate) - translate entire given file using dictionary

* Members:

Dictionary\* dict //pointer to Dictionary object

Class Dictionary which contains:

* Methods:

Dictionary(std::string dictFileName) - create dictionary object from given file, this function assumes that the dictionary file is properly formatted which should be <original word> <translatedWord> for example:

Pies Dog

Kot Cat

std::string translateWord(std::string word) - translate given word and return the translated version (or the original if not found in dictionary)

* Members

std::map<std::string, std::string> m\_wordsList - dictionary is the map

that for given word has its translated version for example m\_worldList["pies"] = dog

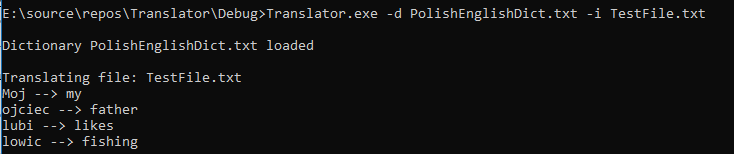
1. Source code

Source code is distributed with program executable, with proper comments of each function.

1. Testing

For testing purpose, the program will print out the translation <original word> --> <translated word>, for the bigger files it is not recommended due to amount of translations.

Working with correct parameters



The content of the input file TestFile is:

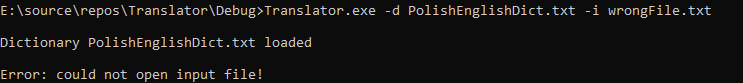
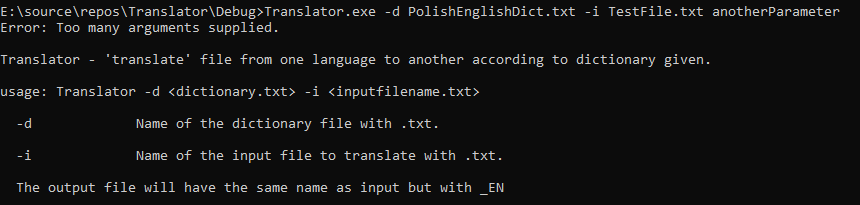
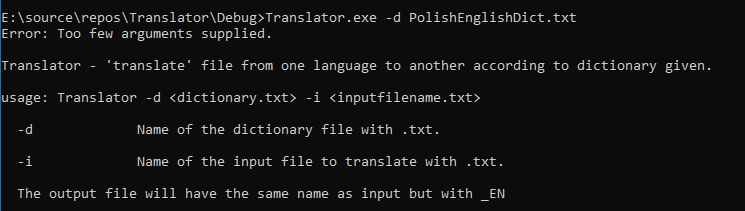
Moj ojciec lubi lowic.

The output of the program

my father likes fishing .

**The program works correctly.**

Working with wrong arguments (too few, too many, wrong input/dictionary file)



1. Conclusions

The program translation is very simple, it does not count the grammar and could change the sense of the sentence.

The program is not error proof in all cases, if the dictionary file is switched with input file the program will fail because input file is not proper formatted.