beCP
2023

Task 2.1: Translation Training (translation)

Author: the beCP team   Preparation: Robin Jadoul

Note: This is an output only *task. This means you only need to submit the results of your computation, and not any kind of code you used. There are several different input files for which you submit solutions together or individually. Of course, this format also means that you are not required or expected to use the same program for each input file, and you could even see if you can solve some things by hand.*

You're training to be a translator so that you can speak the local languages when you go to international programming competitions. The homeworks are generally a lot of effort and you'd rather be training for the programming aspect of the competitions instead. So you usually just hand it all over to a translation program using artificial intelligence.

Disaster strikes! Your internet connection drops, and your homework is due in three hours. Your task is to classify a whole bunch of documents, and indicate for each document in which language it's written, based on a few examples per language.

The list of languages you'll be dealing with is as follows:

- English (`en`)
- Esperanto (`eo`)
- Māori (`mi`)
- Bahasa Indonesia (`id`)
- lojban (`jbo`)

Input

In the download for this task, you're given a set of 5 directories. The first directory, `reference`, contains the known examples per language, named `[language]_[i].txt`. In each of the other directories `set_[subtask]`, you will find some number of files F, simply named `[i].txt`. Each of these files is a document for which you need to determine the correct language.

Output

For each of the four directories, you should submit a single file. The ith line of the file should contain a single word: the (2 or 3-letter) language code of the language for that file. Some skeleton code is provided to deal with reading from the given files.

Scoring

For a set of files (a subtask) with F files to be classified, if you correctly classify x of the F files, you receive $25 \cdot \frac{x}{F}$ points.

Additional constraints

Subtask	Points	Constraints
A	25	$F = 20$
B	25	$F = 100$
C	25	$F = 1\,000$
D	25	$F = 5\,000$

Running your code

In the downloads for this task, you will find some skeleton code `translation.cpp`, where you can fill in code to find the language for some text. You do this by implementing the function `string classify(string)` that takes the content of a file as input, and outputs one of five possible strings corresponding to the language code. A function `bool contains(string haystack, string needle)` is also provided as a further example of something you can do in your code. You are allowed to use this function, but this is not an obligation.

Alongside, you can find a file `driver.cpp`. This file provides the `main` function for the program and it will take care of calling your `classify` function on a single file or on all files in a directory, in the right order.

To compile a program, you can use the following command:
```
g++ -std=c++11 -Wall -Wextra -Wshadow translation.cpp -o translation.
```
To run your code on a single file, for instance `reference/en_1.txt`, you can then issue the command
```
./translation reference/en_1.txt
```
and to run it on every file in e.g. the `set_A` directory:
`./translation set_A`. These will output the languages your program detects to the screen, and warn if any of them are not valid language codes. To save the output to a file so that you can submit it, you can append `> filename_to_save.txt` to the end of these commands. Like this:
```
./translation set_A > set_A_solutions.txt.
```

If you want to print any debugging information, make sure to use `cerr` instead of `cout`.

You are allowed to ask for technical help in getting your code running and executing on the task's data by asking an attendant or sending a question through CMS.