BBC Technical Challenge

by Steven Blowers

Approach

The instructions stated the solution should be submitted in a runnable format, I therefore decided to use a compiled language which would allow me to submit an executable file that could be run without any dependencies.

From the compiled languages on offer, I wanted a language that supported user input and output from the command-line, had a good built-in unit testing framework and would allow my solution to run fast, I therefore settled on Rust.

In terms of how my roman numerals encoder would work, my initial ideas were around building up the result from left to right, whilst keeping track of the remaining decimal number. After a quick Google search it seemed that basic algorithm had potential, so I went forward with it.

After getting my encoder to work, with test coverage, I then built up the code around it to allow the user to input a number as an argument when running the executable.

At the beginning of writing the command-line interaction part of the code, I decided test coverage was not neccesary as it did not contain any business logic and if tests were written they would mostly be testing functionality provided by the standard library. However, after having written the code I found there was a section of code, around parsing the command-line arguments, that I did want coverage of. So, I decided to refactor that part out to its own module and add test coverage around it.

Assumptions made

The input/output will be provided via the command-line.

Caveats

This is the most complex project I have completed in the Rust programming language. It does work, however there are likely parts of the code which are not idiomatic Rust code.