Overview

The objective of this project is to create a working calculator divided into 4 different programs which will work in a sequence to obtain the result (meaning the output from the first program will be the input of the second one and so on). The first one will consist of a tokenizer which will split the input from the user into tokens (integers, floating point numbers, operators...). Then, the infix-to-postfix translator will change the notation to a simpler one, to facilitate the creation of the code in the next step (code generator) which will translate the notation to the instructions, which will be executed by the final program, the Virtual Machine. All these programs will be written in C, without the use of any library apart from the ones included in the language.

Requirement

General requirements:

* The calculator should always return the correct answer
* The calculator should calculate with these operators: +, -, \*, / ,% ,^ and ( )
* The calculator should work with floating-point and integer numbers
* The calculations should be finished in less than 1 seconds
* The calculator should take the input as file (textfile)
* The calculator should be able to calculate 10 different results without crashing
* The calculator shall support only one inputfile at once and must report the result before taking another input
* The calculator shall only start calculating when the input consists of the operators specified above and integer or floating-point number, otherwise there shall be an Error messages
* The inputfile must have a normal arithmetic syntax, otherwise the calculator shall return an Error message
* The calculator should consist of four parts: Tokenizer, the infix-to-postfix translator, code generator and the virtual machine
* Every part and the calculator should be able to be tested with a test that is written in under 1 minute

Requirements for the Tokenizer

* The tokenizer shall be able to process the inputfile and return a file where all operators and numbers are separated by a new line sign (\n)
* The tokenizer shall process the operators specified above and integer or floating-point numbers
* For every number or operator there should be the “value” (e.g. 1 or +) and the type identifier (e.g. int or op (for operator))

Requirements for the Infix-to-postfix translator

* The translator shall process the file from the tokenizer and returns a file where the operators and numbers are in the postfix notation
* The translator shall be able to process the “value” and the type identifier

Requirements for the code generator

* The generator shall process the file from the translator and return a file with instructions
* The generator shall be able to process the operators and numbers
* The generator shall translate the numbers and operators to a given instruction set

Requirements for the virtual machine (vm)

* The vm shall process the file from the generator and return an integer or floating-point number
* The vm shall be able to process the instructions and execute the instructions