Software Requirements of a Calculator

Authors: Team CS3500.2020.X4

Yang Chen - 119100224
Tingting Xun - 118100140
Kevin Smith - 119111858
Melanie Abercrombie - 119111737

Overview:

This project upon completion will function as a simple scientific calculator. Coded using C, this calculator will be able to implement addition, subtraction, multiplication, division, exponential functions and so on. This system will take user input, differentiate between digits and operation symbols, apply the proper algorithm, and output a correct answer. It will check the legality of the input, including filtering all illegal characters, checking for matched parentheses, and verifying the proper number of operands for every function. This will be implemented with a stack data structure, which will also ensure the proper order of operations.

Requirements :

This calculator must be able to interpret a user inputted infix expression and output the correct simplified solution.
This calculator must notify the user in the event of erroneous user input.
This calculator must have an intuitive user experience with concise user manual.
This calculator must perform calculations quickly and correctly.

The supported operations must include the following basic functions:
Addition, Subtraction, Multiplication, Division, Modulus Division, Parentheses, Exponent

Support Basic Functions:

- Addition
  The addition operator is used by inputting "+".
  Example: a+b

- Subtraction
  The subtraction operator is used by inputting "-".
  Example: a-b

- Multiplication
  The multiplication operator is used by inputting "*".
  Example: a*b.

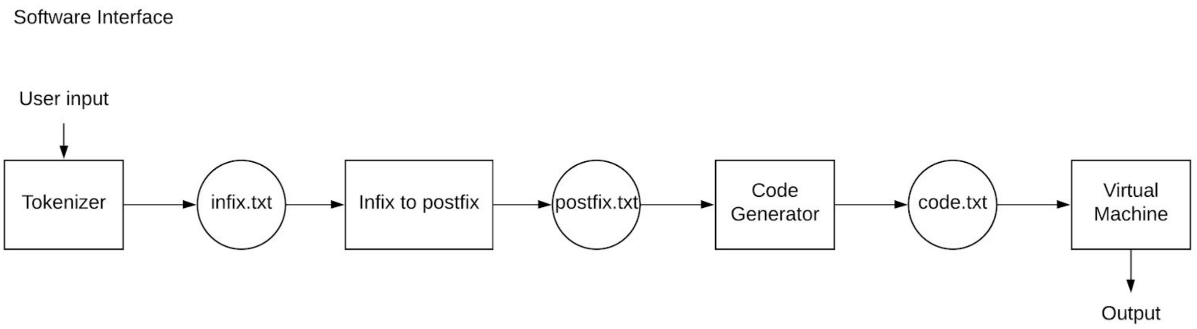
- Division
  The division operator is used by inputting "/" .
  Example: a/b.

- Modulus Division
  The modulo operator is used by inputting “%”.
  Example: a%b
- Parenthesis
  The parentheses operators are used by inputting “(“ and “)”
  Example: a*(b+c)
- Exponent
  The exponent operator is used by inputting “^”.
  Example: a^b

This calculator will be modularized into four main components:

- Tokenizer
- Infixtopostfix (I2P) converter
- Code Generator
- Interpreter

Interfaces Description :



- Four interface: Tokenizer, Infix2postfix converter , Code generator, Interpreter/virtual machine

1. Tokenizer

   The tokenizer distinguishes between the various "tokens" from the user's input. For this project, "tokens" describe characters representing supported integers and operators.

2. Infix2postfix converter

   The infix2postfix converter takes the output from the tokenizer that is in infix format and outputs a postfix expression.

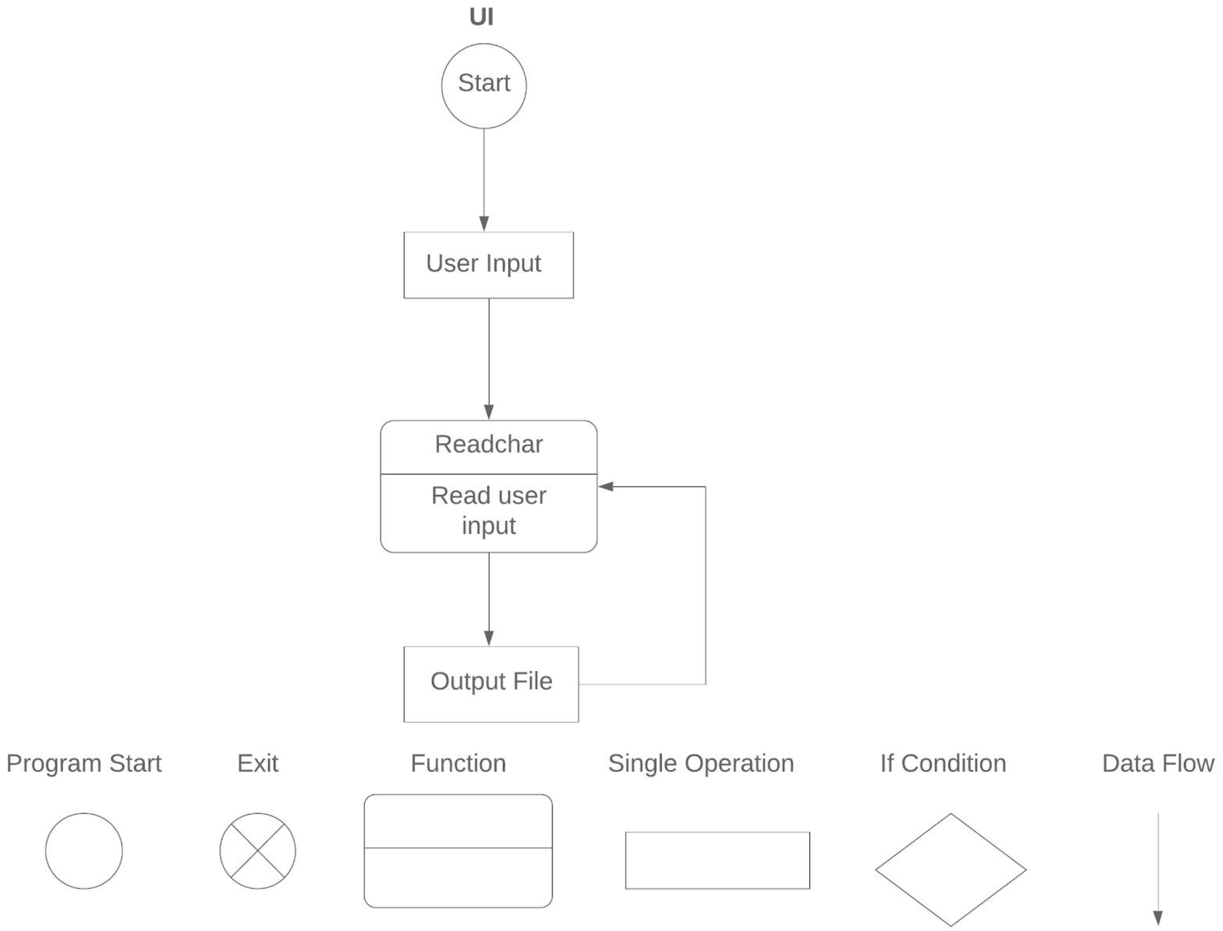
3. Code generator

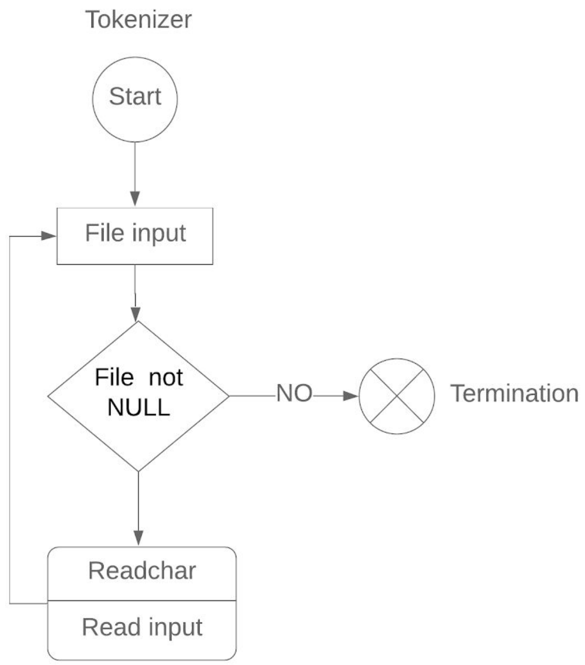
   The code generator analyzes the postfix expression and pushes instructions onto a stack such that the instructions will be popped from the stack in the appropriate sequential order for the calculation of the postfix expression.

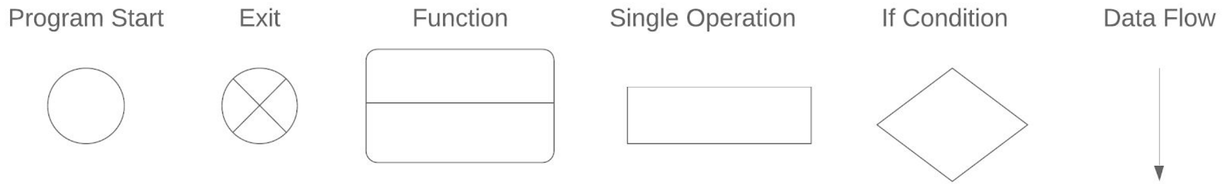
4. Interpreter/virtual machine

   The interpreter receives a stack of instructions from the code generator. Instructions are popped and executed from the stack until the stack is empty. The interpreter outputs the simplified solution.

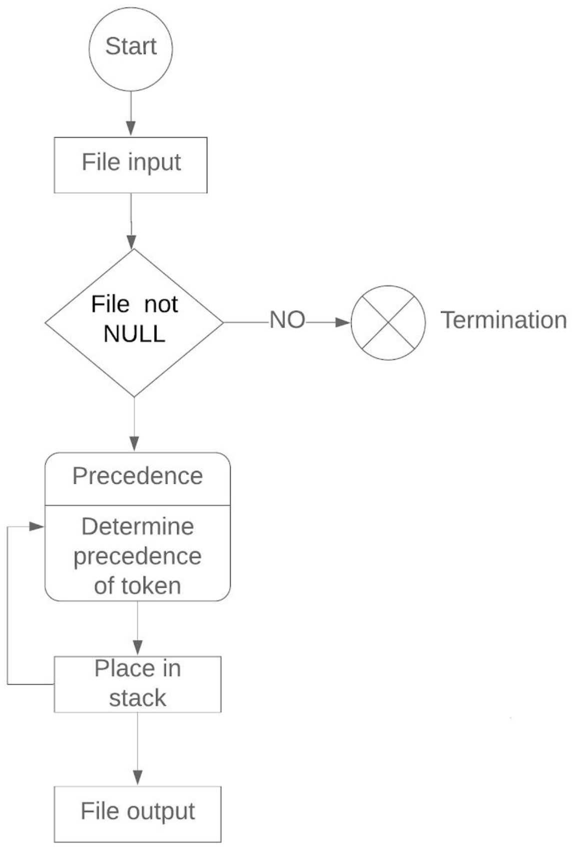
Data Flow Diagram & high level architecture diagram:

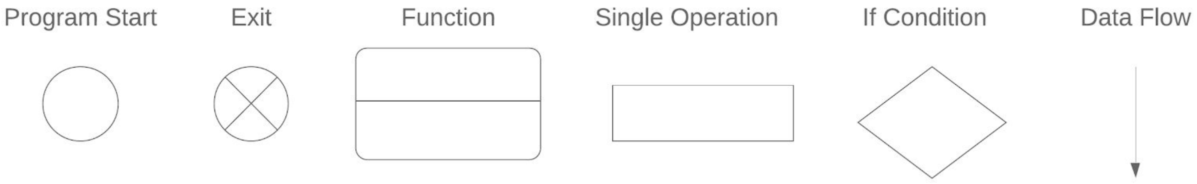



Tokenizer
Start
File input
File not NULL
NO
Termination
Readchar
Read input

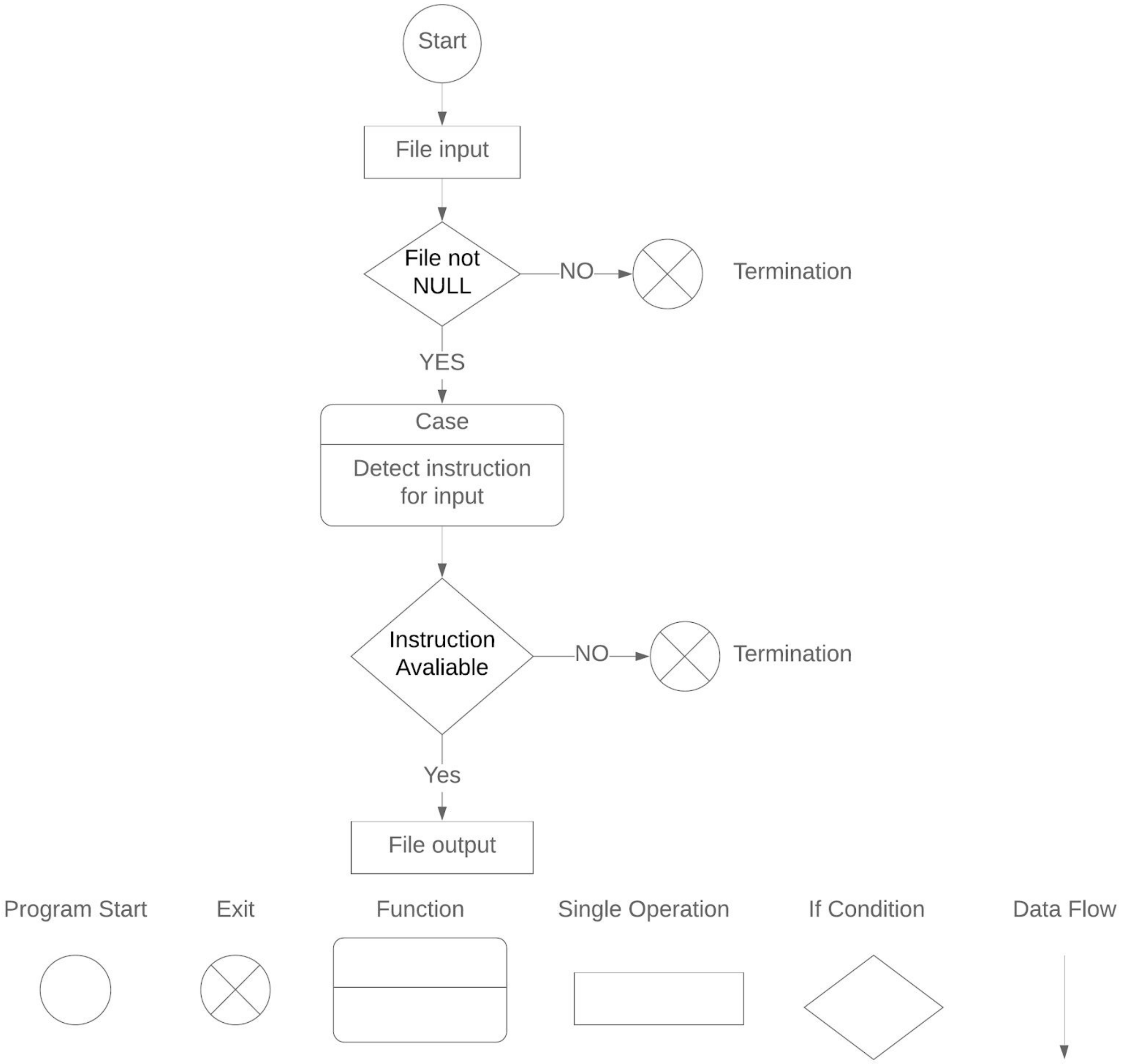

Program Start
Exit
Function
Single Operation
If Condition
Data Flow

Infix to Postfix

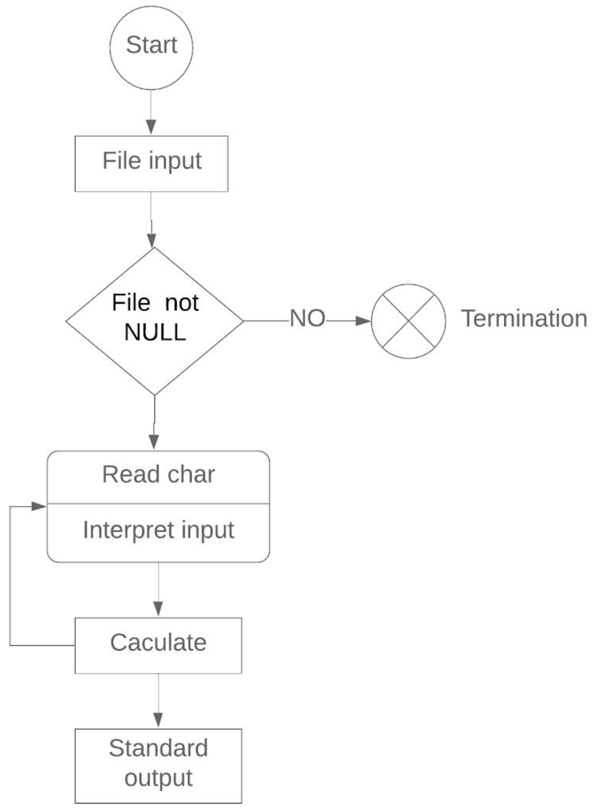


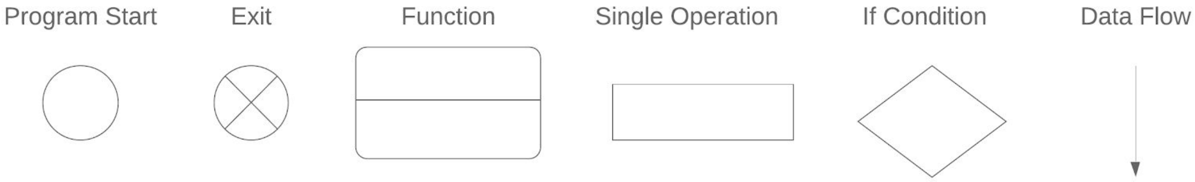


Code Generator

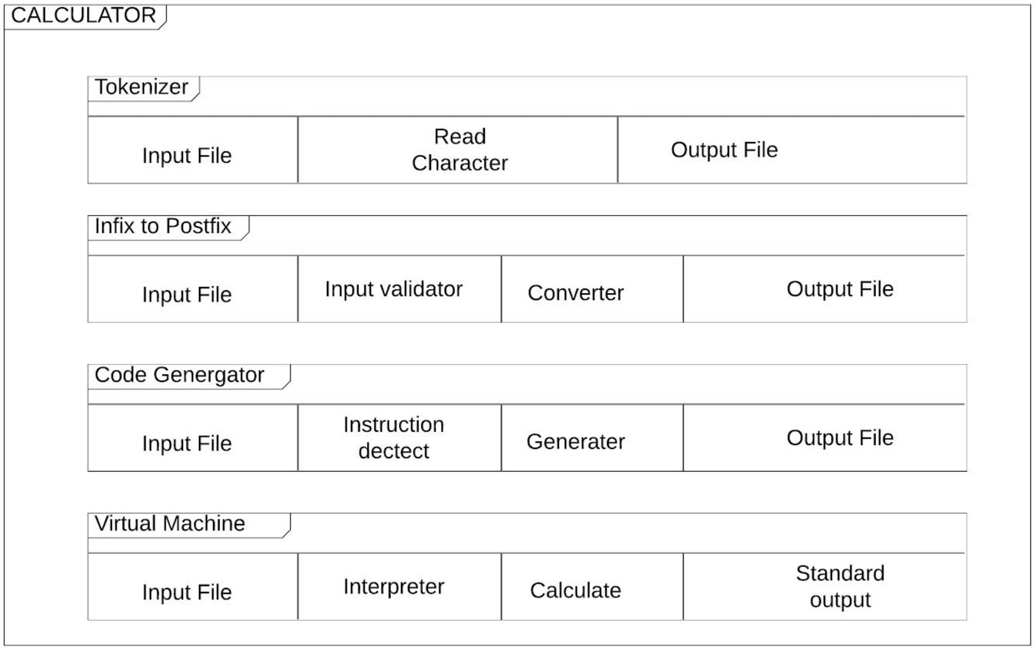


Interpreter





SOFTWARE ARCHITECTURE



CALCULATOR

Tokenizer

Input File	Read Character	Output File

Infix to Postfix

Input File	Input validator	Converter	Output File

Code Genergator

Input File	Instruction dectect	Generater	Output File

Virtual Machine

Input File	Interpreter	Calculate	Standard output