Late Nighterz

<Kool Kalculator>
Software Architecture Document

Version 1.0

Revision History

Date	Version	Description	Author
11/09/23	1.0	Started and finished software architecture document	David Donaldson, Ian Collins, Noah O'Grady, Ky Jost, Humzeh Al-Tamari

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

The purpose of this document is to outline the overall structure of the software. It explains how the end user will interact with the software, including all user interfaces, and how we as a team will build the software. It provides an architectural overview of the system. All significant architectural choices are recorded in this document.

1.2 Scope

This document influences software components, connections between software, configuration of software, and rationale for architectural design.

1.3 Definitions, Acronyms, and Abbreviations

Definitions:

- **Mathematic Operator:**
  - Symbol used for specifying relationship and transformation of numbers
- **Order of Operations:**
  - The specific order in which arithmetic statements are solved given their operation and position.
- **Arithmetic:**
  - Mathematical operations that perform work upon real numbers of the real number line. One of the oldest forms of math.
- **Addition (ADD)**:
  - Operation that combines two numbers to find their sum.
- **Subtraction (SUB)**:
  - Operation that finds the difference between two numbers.
- **Multiplication (MUL)**:
  - Operation that calculates the product of two numbers.
- **Division (DIV)**:
  - Operation that divides one number by another to find the quotient.
- **Exponentiation (POW)**:
  - Operation that raises a base number to a specified power.
- **Parentheses (BRACKETS)**:
  - Symbols "(" and ")" used to group expressions and control the order of operations.
- **Syntax Error**:
  - An error that occurs when the input expression is not formatted correctly or violates the rules of the calculator's syntax.
- **Uncalculatable Error**:
  - An error that occurs when the calculator cannot perform a calculation, often due to dividing by zero or taking the square root of a negative number.

Acronyms:

- **PEMDAS:**
  - Parenthesis, Exponible, Multiplication, Division, Addition, Subtraction

- **GUI:**
  - Graphical User Interface

1.4 References

The project requirements provided by the instructor will dictate large amounts of the functional requirements for this project.

1.5 Overview

The following pages will contain the architectural representation of the software, the architectural goals and constraints, the logical view of our software, the interface description, and the ways in which the architecture will contribute to the quality of the software.

2. Architectural Representation

- **Conceptual View:**
  Takes in mathematical problems and either identifies errors or offers results.

- **Component and Connector View:**
  Components:
  -HTML UI
  -C++ Evaluator
  Connectors:
  -JS

  UI will take in input, send input to text file. C++ program will read contents, parse it, and send results back to file. UI will retrieve results from the file.

- **Data View:**
  Calculates one equation at a time, does not store any information.

- **Pattern View:**
  Errors are handled by separate functions before the operations take place. If any errors occur, the software will store those messages and output them to the UI. If no errors are detected, the normal order of operations will occur and output the result of the statement.

3. Architectural Goals and Constraints

Architectural Goals:

- C++ program is Object Oriented
- The program will be reuseable. We will utilize JavaScript in the HTML file to accomplish this. Once the C++ file sends the output to the JavaScript, it will edit the HTML DOM to reflect the output. The JavaScript can continually change the DOM, so it will be able to run the program over and over, and previous iterations will not affect the current one.
- The C++ program will be highly portable. We will use JavaScript to accomplish this. The C++ file will be compiled, and the JavaScript will be what calls the C++ file. This means that it will not matter what system the program is run on, as the JavaScript file is sending the input/receiving the output from the C++ file.

- Our program does not have many security concerns, as the user should not be inputting any sensitive information. However, if they do, the program does not save previous entries, so any user who decides to input their social security number will be safe. This will also alleviate privacy concerns.

Architectural Constraints:

- Expression evaluator must be coded in C++: To create a website UI, we must export input from HTML file to C++ function to handle execution and return result to HTML file. We will accomplish this by using JavaScript written in the HTML file. The JavaScript will send the input to the C++ file. Upon completion, the C++ file will send the output to the JavaScript file. The JavaScript file will then print the output onto the HTML webpage.
- The program must be completed by 03 December. This includes the testing and user manual.

4. Logical View

4.1 Overview

The overall software divides into the front-end GUI and the back-end computation of the arithmetic expression. The GUI will pass the statement to the back end where the program returns either an error message or the result of the statement. The program will identify any errors before the computation occurs by passing the error message back to the front-end. If no errors are detected, the result of the expression is returned to the front-end and displayed. Each error check is run separately as a function, and each error is logged in a list. That list is passed to the front-end in the event of an error. The order of operations function is run if no errors are detected.

4.2 Architecturally Significant Design Modules or Packages

- C++ Expression Evaluator – Will parse given equation, check for errors, return error if there is one, otherwise return result
    - Math module: Takes a valid expression and solves it according to the order of operations.
        - Parentheses
        - Exponents
        - Multiplication
        - Division
        - Addition
        - Subtraction
    - Error Handling: Determines if the expression is valid or not. If valid, passes to math module. If not, returns a list of errors in the expression.
        - Mismatched Parentheses: The expression does not have an equal number of ‘(‘ and ‘)’ parentheses
        - Operators without Operands: The expression contains an operator that does not have operands on both sides
        - Invalid Operator Usage: The answer to the expression is undefined in mathematics (divide by zero, square root of a negative)
        - Missing Operator: The expression is missing an operator between an operand and a parentheses
        - Invalid Characters: The expression contains characters such as letters or special

characters that are not part of any mathematical expression

- HTML/JavaScript UI Design – Where user interfaces with the input/output
    - Input: HTML form will take user input. JavaScript will pass user input to C++ file.
    - Output: C++ file will pass output to JavaScript. JavaScript will edit HTML DOM and print the output on the webpage.

5. Interface Description

We will be creating a website as our user interface for the program. As such, our user interface will be built in html and JavaScript. The interface will consist of a form the user can enter input to. Upon pressing the submit button, the value in the form will be passed to the C++ program to be calculated or rejected.

Valid inputs will be inputs that do not include the following errors:

- Mismatched Parentheses: The expression does not have an equal number of '(' and ')' parentheses
- Operators without Operands: The expression contains an operator that does not have operands on both sides
- Invalid Operator Usage: The answer to the expression is undefined in mathematics (divide by zero, square root of a negative)
- Missing Operator: The expression is missing an operator between an operand and a parentheses
- Invalid Characters: The expression contains characters such as letters or special characters that are not part of any mathematical expression

Outputs will be printed onto the webpage. The C++ file will pass either the result or a list of errors to the JavaScript file. From here, the answer will be printed in the following format.

ANSWER: '*answer'*

Errors will be printed in this format:

ERROR: '*list of errors'*

The list of errors will contain every error the expression contains.

6. Quality

The software architecture contributes to extensibility by being a relatively simple system that won't be affected by outside programs changing. Once the program is complete, the code will be static. However, if we decide to change the functionality later, it will be possible to do so. Our program will contain functions that determine errors and order of operations. Due to this structure, if we want to change the program later, we just need to change one function, or add a new function, which is easier than having to edit the entire program. The software architecture contributes to reliability by being a static program without much input from outside files. It is contained within our C++, JavaScript, and html files and the only parts outside our control are the libraries we input, which are built to be used in such a way. Our program will be portable. We have two files, a C++ file and a JavaScript/html file. The C++ file will be compiled, and it will run through the JavaScript/html file. Once the C++ is complete and compiled, if the two files are kept in the same place, the program will be able to run anywhere. Our program is also compatible with other programs. While it does not have interoperability with other programs, it is very good at co-existence. Our program's presence on a system will not interfere with the operation of programs the system is running. Lastly our program will be very easy to maintain since the core component of it will not change.