KU EECS 348

Arithmetic Expression Evaluator in C++
Software Requirements Specifications

Version 1.0

Revision History

Date	Version	Description	Author
10/9/2023	1.0	The original version of this document.	Hannah

Table of Contents

Software Requirements Specifications

1. Introduction

1.1 Purpose

The Software Requirements Specifications document is a document that describes what and how the software will perform. It also takes stakeholders' needs into account and describes the functionality needed to fulfill these needs. Finally, the Software Requirements Specifications document outlines nonfunctional requirements, design constraints, and anything else needed to give a complete description of the software requirements.

1.2 Scope

This Software Requirements Specifications document applies to the software developed by this group for KU's EECS 348 class. It will apply only to the calculator project developed by this group.

1.3 Definitions, Acronyms, and Abbreviations

Data structures: Various methods used to sore and organize data
Heap: Data structure specialization of a complete binary tree
PEMDAS: Indicates order of operations – parenthesis, exponents, multiplication, division, addition, subtraction
Error handling: How a group recovers from an error found in the software
Debugging: Process of removing errors from software
Object-oriented programming: Language model that organizes design around objects such as classes or data fields
SRS: Software Requirements Specifications – a document to determine the requirements of the given project

1.4 References

EECS348: Term Project in C++, 09/07/2023, University of Kansas

Arithmetic Expression Evaluator in C++ Software Development Plan, 09/24/2023, University of Kansas

1.5 Overview

The rest of the SRS will contain an overall description, the specific requirements, a classification of the functional requirements and an appendix. The overall description section will pertain to the factors that affect the product and its requirements, the specific requirements section will include a more comprehensive description of the actual requirements, and the classification of requirements will list the requirements by type: essential, desirable, and optional.

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

The system interface will apply data structures (stack or tree) to parse the expression. It will then use the rules of arithmetic operators to evaluate the expression and handle errors. Unit tests will also be included in the system interfaces.

2.1.2 User Interfaces

This project uses command line interface (CLI), where the input is received as an arithmetic expression and the output is displayed, along with any error messages. The user would also have access to a user manual or README file to explain how to interact with the program along with examples.

2.1.3 Hardware Interfaces

There wouldn't be many hardware interfaces since the arithmetic evaluator is a software program. However, hardware components would be needed like a computing device with storage and memory as well as a display for outputs.

2.1.4 Software Interfaces

C++ programming language is used as a software interface to create source code and logic. Development tools may be used like visual studio code or other code editors. Linux as the operating system will be used as well as the command line interface. Basic C++ libraries could also be used.

2.1.5 Communication Interfaces

The user-program interaction will be through the command line interface. The user will give an input after looking at instructions in the manual and the system will either give an output as an answer or give an error message to convey any problems with the user input.

2.1.6 Memory Constraints

Memory is required for the system to handle the length of the arithmetic expression being input by the user. It would also be used to store each node/element in the data structure used for parsing (stack or tree).

2.1.7 Operations

The operations necessary in this program are input parsing, operations/parentheses precedence handling, error handling, and handling multiple expressions.

2.2 Product functions

The product should be able to calculate functions that include *, /, +, -, (,), %, ^, and numeric constants. This should be able to follow the order of operations (PEMDAS).

2.3 User characteristics

All users of this product should have a firm grasp on basic mathematical order of operations (and how parenthesis are used in this structure) including addition, subtraction, multiplication, division, and exponents. Additionally, they need an understanding of more complex concepts, such as modulus. Finally, they must be able to utilize a text-based interface to appropriately use the product.

2.4 Constraints

The constraints for the development process of the project include that it will be written in C++, and it must be compiled on a Linux computer.

2.5 Assumptions and dependencies

The assumption that exists in this SRS is that each member will have access to a machine capable of compiling C++.

2.6 Requirements subsets

Requirement subsets that exist are functional and non-functional. Functional requirements are requirements that the user will interact with. Non-functional requirements are needs that are set by extraneous circumstances. There will also be requirements that are specific to the user menu and other requirements will be more focused on data storage and prioritization. Additionally, the functional requirements will be categorized into desirable and essential. Essential meaning that it has to be there and desirable meaning that we would like to add it if possible.

3. Specific Requirements

3.1 Functionality

3.1.1 Expression Parsing

Program should be able to parse arithmetic expressions entered by the user.

3.1.2 Operator Support

Program should implement the operators: + (addition), - (subtraction), * (multiplication), / (division), % (modulo), and ^ (exponentiation). The ** operator can optionally be allowed as an alternative to ^.

3.1.3 Operator Precedence

Program should define precedence of the operators according to the PEMDAS rules.

3.1.4 Parenthesis Handling

Program should identify and evaluate expressions within parentheses and use parentheses to determine the order of evaluation.

3.1.5 Numeric Constants

Program should recognize and calculate numeric constants within the expression. Assume input will be integers. Support for floating point numbers may also be implemented.

3.1.6 User Interface

Program should have a user-friendly and legible command-line interface that allows users to enter expressions and displays the calculated results

3.1.7 Error Handling

Program should have robust error handling to manage scenarios like division by zero or invalid expressions.

3.2 Use-Case Specifications

User should be able to enter an arithmetic equation and get the accurate output. The software will limit the user to addition, subtraction, multiplication, division, modulo, and exponential. The program will follow the rules of PEMDAS and be able to use parenthesis handling. The software will have a user-friendly interface and be able to handle any input the user gives it. All of the coding will be completed in C++.

3.3 Supplementary Requirements

Additional requirements include the necessity of a text-based user interface, using object-oriented principles, and the usage of comments/other documentation. These are necessary to fulfill all other aspects of the project deliverables.

4. Classification of Functional Requirements

Functionality	Type
Expression Parsing	Essential
Operator Support (+, -, *, /, %, ^)	Essential
Operator Support (**)	Desirable
Operator Precedence	Essential
Parenthesis Handling	Essential
Numeric Constants (Integers)	Essential
Numeric Constants (Floating Point Numbers)	Desirable
User Interface	Essential
Error Handling	Essential

5. Appendices

We will create a user manual to assist the user in understanding how the user will interact with the calculator. This would be a part of the requirements.