Kansas Instruments

Arithmetic Expression Evaluator
Software Requirements Specifications

Version 1.08

Revision History

Date	Version	Description	Author
10/10/24	1.0	Role distributing	Phoenix, Emma, Aiden, Jakob, Khang
10/13/24	1.01	Completed 2.1.7-2.4 and added to 5	Phoenix
10/13/24	1.02	Completed 2.1.1-2.1.4 and added to 5	Emma
10/16/24	1.03	Added to 3.2, 3.3, and 5	Khang
10/18/24	1.04	Completed all of section 1	Kit
10/19/24	1.05	Completed sections 3.1 and 3.1.1	Aiden
10/19/24	1.06	Completed all of section 4	Jakob
10/19/24	1.07	Reviewed and completed sections 3.2, 3.3	Khang
10/20/24	1.08	Added and finalized sections 2.1.5-6, 2.5, and 2.6	Vrishank

Table of Contents

Software Requirements Specifications

1. Introduction

1.1 Purpose

The purpose of this SRS is to define the requirements for developing an Arithmetic Expression Evaluator using C++. This evaluator will allow users to input arithmetic expressions and provide the correct result by following the order of operations (PEMDAS). This document describes both the functional and non-functional requirements, design constraints, and necessary specifications for developing the evaluator. It will serve as the basis for guiding the development process and ensuring that the final product meets the outlined requirements.

1.2 Scope

This SRS applies to the Arithmetic Expression Evaluator project and outlines its features, including parsing user input for arithmetic expressions, supporting operators (addition, subtraction, multiplication, etc.), and handling errors for invalid inputs or operations. The evaluator will support both integer and possible floating-point operations, along with error messages for invalid expressions. This document will also cover the project's constraints and deliverables as defined in the Software Development Plan(Project-Plan 1).

1.3 Definitions, Acronyms, and Abbreviations

PEMDAS: Parentheses, Exponents, Multiplication/Division, Addition/Subtraction; the rule used to define operator precedence in arithmetic expressions.

1.4 References

Project plan 1

EECS 348 Term project

- https://makingofsoftware.com/2020/02/product-requirements-specification-product-perspective/
- https://cs.mtech.edu/main/images/standards/programProjects/mtmprogprodspptmplt.docx
- https://argondigital.com/resource/whitepapers/everything-you-wanted-to-know-about-interface-requirements/
- Use Case Diagrams: An Introduction - Altkom Software.
- https://www.indeed.com/career-advice/career-development/list-of-use-cases-examples

1.5 Overview

Our SRS outlines the requirements for the Arithmetic Expression Evaluator project. It includes the purpose, scope, and key definitions. Our document provides an overall description of system interfaces, user interactions via a command-line interface and core functionalities such as parsing arithmetic expressions and handling errors. It also details the specific functional requirements, use cases, and supplementary specifications like accessibility and integration. Lastly, the appendices offer supporting references and additional documentation.

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

Our arithmetic expression evaluator will not require any external system interfaces in its initial version. In the future, we could involve interaction with a compiler system where the evaluator might be used as a component for parsing and evaluating expressions as part of a larger compilation process.

2.1.2 User Interfaces

The user will interact with the evaluator through a command-line interface (CLI). The CLI will prompt the user to input various arithmetic expressions using the +, -, *, /, %, and ** operators along with numeric constants and parentheses. The program will display the user the calculated result or an error message if the input is invalid.

2.1.3 Hardware Interfaces

Our arithmetic expression evaluator will not require any specialized hardware. It will be able to run on standard desktop or laptop computers. The program only requires a basic CPU for computation and standard keyboard input for user interaction.

2.1.4 Software Interfaces

The program will be developed in C++ and compiled using standard C++ compilers. The evaluator will make use of the C++ Standard Library for operations like input/output and string manipulation. We currently do not require any additional third-party libraries.

2.1.5 Communication Interfaces

Our arithmetic expression parser does not have specific communication interfaces and will not interact with external databases or servers.

2.1.6 Memory Constraints

The product we will deliver is expected to run efficiently within the memory constraints of typical desktop and laptop computers. The constraints will be determined by the machine that the product is run on. The inputs will the constrained to a certain number of bits. However, it is unlikely that the user will access the full number of bits that the program allows for as it could be 16, 32, or more bits.

2.1.7 Operations

Our software's operations include taking in a user’s arithmetic input and evaluating it. The mathematical operators our software should handle include multiplication, division, addition, subtraction, exponentiation, and the modulo operator, as well as taking into account the precedence of parentheses.

2.2 Product functions

- Be able to evaluate mathematical input
- Handle various operator types (multiplication, division, addition, subtraction, exponentiation, and the modulo operator)
- Handle parentheses
- Contains a user interface for inputting expressions
- Check for nonsensical inputs or mathematically impossible expressions (such as division by 0)

2.3 User characteristics

The types of users who will be using our program will be TAs and/or the course the professor.

Characteristics include:

Strenuous testing and use cases to see what the program will handle
Evaluation of program structure and documentation
Willing to use and follow a readme

2.4 Constraints

The constraints we will be facing:

Follow the laws and rules of math and mathematical evaluation
Be able to support operation on both computers of the developers, and computers of the users.
Create a data structure that stores the expression's structure

2.5 Assumptions and dependencies

The following assumptions and dependencies are outlined for the development of the Arithmetic Expression Evaluator:

- Assumes access to a C++ compiler.
- All necessary functions can be achieved through the C++ Standard Library.
- A machine that can run the final product

2.6 Requirements subsets

Some essential requirements are basic arithmetic operations, error handling, and command-line interface implementation. Optional requirements such as floating-point support and advanced parentheses handling are considered for future development.

3. Specific Requirements

3.1 Functionality

The evaluator should follow PEMDAS rules to ensure that expressions are evaluated properly, respecting operator precedence (parentheses, exponents, multiplication/division, addition/subtraction). This software will support the following operators: + (addition), - (subtraction), * (multiplication), / (division), % (modulus), and ** (exponentiation). To prevent issues during runtime, errors should be identified (e.g. such as mismatched parenthesis, division by zero) and resolved by issuing an alert to the actor (e.g. the user). For this project, stacks will be the optimal data structure to parse and evaluate data, especially when handling properly closed parentheses. For a Command-Line Interface (CLI), the evaluator should provide basic instructions to prompt the user to enter expressions accompanied by the resulting outcome or alert due to improper input.

3.1.1 <Functional Requirement One>

The software should be capable of parsing a given arithmetic expression according to operator precedence and representing it internally for evaluation.

3.2 Use-Case Specifications

Name: Evaluate arithmetic expression

Description: Parse through and evaluate a given arithmetic expression according to PEMDAS

Actor: Developers, users, larger compiler product

Trigger: Arithmetic expression inputted

Pre-conditions: The arithmetic expression makes use of supported operations addition, subtraction, multiplication, division, modulo, or exponentiation.

Basic-flow: A user uses the compiler product and an arithmetic expression is found in their source code. The compiler passed on the arithmetic expression to the evaluator component. The program then parses through the arithmetic expression and represents it through a stack or tree, establishes the order of operations if parentheses are used, makes use of supported operators following PEMDAS, and returns the

result of the arithmetic expression

Alternative path: If an error is found such as an odd number of parentheses, division by 0, etc., the program handles it and issues a warning to the user. specifying the type of error found.

3.3 Supplementary Requirements

- Readability with comments and documentation, allowing easy modification by future developers
- Accessibility, easy to use
- Well-structured to be able to incorporate into a larger compiler product without issues
- Reliable, consistent results that pass rigorous testing for different inputs

4. Classification of Functional Requirements

Functionality	Type
Handle operator precedence (PEMDAS)	Essential
Support the following operators: +, -, *, /, %, **	Essential
Handle parentheses for precedence and grouping	Essential
Evaluate numeric constants (initially integers only)	Essential
Provide a command-line interface for input/output	Essential
Error handling for division by zero	Essential
Data structure to represent the expression (stack/tree)	Desirable
Provide informative error messages	Desirable
Provide user manual or README	Desirable
Recognize invalid expressions	Desirable
Recognize nested parentheses	Essential
Handle extraneous parentheses	Optional
Support floating-point numbers	Optional

5. Appendices

No items included in this appendix are a requirement for the documentation of this program or the program itself

References Used In Completion of This Software and Any Documentation Pertaining To It

https://makingofsoftware.com/2020/02/product-requirements-specification-product-perspective/

https://cs.mtech.edu/main/images/standards/programProjects/mtmprogprodspptmplt.docx

https://argondigital.com/resource/whitepapers/everything-you-wanted-to-know-about-interface-requirements/

Use Case Diagrams: An Introduction - Altkom Software.

https://www.indeed.com/career-advice/career-development/list-of-use-cases-examples

https://www.jamasoftware.com/requirements-management-guide/writing-requirements/functional-requirements-examples-and-templates

https://bettercli.org/