Foobar Inc

# C++ Arithmetic Evaluator
# Software Requirements Specifications

**Version 0.2**

# Revision History

| Date | Version | Description | Author |
| --- | --- | --- | --- |
| 14/Oct/2023 | 0.1 | Initial Project Specification | Cody Duong |
| 14/Oct/2023 | 0.2 | Change dd/mm/yyyy formatting to dd/mmm/yyyy formatting in header | Cody Duong |
| | | | |
| | | | |

# Table of Contents

# Software Requirements Specifications

## 1. Introduction

The purpose of this Software Requirements Specification (**SRS**) is to describe requirements and more for the C++ Arithmetic Evaluator (**the project)**. **The project**'s specific description, timeline, and deadlines can be found in **the project**'s Software Development Plan (**SDP**), see *Section 1.3 Definitions, Acronyms, and Abbreviations* and/or *Section 1.4 References*.

This **SRS** describes all systems and subsystems of **the project**. The **SRS** describes **the project**'s context as well as specific product perspectives. It also describes constraints/assumptions, as well as the derived core and non-core requirements.

### 1.1 Purpose

The purpose of the **SRS** is to describe **the project**. **The project** should be a C++ program can parse and evaluate arithmetic expressions containing the following operators:

- addition (+)
- subtraction (-)
- multiplication (*)
- division (/)
- modulus (%)
- exponentiation (^)

The program should also be able to handle expressions with parentheses to define precedence and grouping. It is essential that numeric constants are supported, with float constants as desirable. Algebraic notation/substitution is not required (i.e., alphabetic constants).

### 1.2 Scope

This SRS describes all systems and subsystems of **the project**. It describes in its entirety the expected functional and non-functional requirements for the entirety of the project.

### 1.3 Definitions, Acronyms, and Abbreviations

**The project/this project/the program**: "C++ Arithmetic Evaluator", described by the Software Development Plan (**SDS**). The repository for all code and documentation lives on GitHub, at https://github.com/codyduong/EECS-348-Project/.

**SRS**: Software Requirements Specification. See *Section 1.4 References - SRS.*

**SDP**: Software Development Plan. See *Section 1.4 References – SDP*.

**Functional Requirement**: Program functionality that requires code modification to meet the requirement.

**Supplemental Requirement**: Program documentation or otherwise that does not require code modification but can improve useability and usefulness of program.

### 1.4 References

**SRS**: Available at https://github.com/codyduong/EECS-348-Project/

**SDP**: Available at https://github.com/codyduong/EECS-348-Project/

### 1.5 Overview

The rest of the document describes an overall description, product perspectives—assumptions, user characteristics, interfaces—and functional/non-functional requirements.

# Overall Description

The arithmetic expression parser will be a command-line program that takes an arithmetic expression as input and outputs the result of evaluating the expression. The program will be written in C++ and will use a variety of programming techniques, possibly including recursion, stacks, linked lists, and more.

## 1.6 Product perspective

### *1.6.1 System Interfaces*

The arithmetic expression parser will not have any system interfaces.

### *1.6.2 User Interfaces*

The arithmetic expression parser will have a simple command-line user interface. The user will be able to enter an arithmetic expression as input and the program will output the result of evaluating the expression.

### *1.6.3 Hardware Interfaces*

The arithmetic expression parser will not have any hardware interfaces.

### *1.6.4 Software Interfaces*

The arithmetic expression parser will not have any software interfaces. *Note: Perhaps it may be useful for testing purposes to expose a Software Interface?*

### *1.6.5 Communication Interfaces*

The arithmetic expression parser will not have any communication interfaces.

### *1.6.6 Memory Constraints*

The arithmetic expression parser will have minimal memory constraints. The program will only need to store the arithmetic expression being parsed and the result of evaluating the expression.

### *1.6.7 Operations*

The arithmetic expression parser will be able to perform the following operations:

- Parse arithmetic expressions
- Evaluate arithmetic expressions

## 1.7 Product functions

**The program** will be able to parse and evaluate arithmetic expressions containing the following operators +, -, *, /, %, and ^ as well as numeric constants. The program should be able to handle expressions with parentheses to define precedence and grouping. It is essential that numeric constants are supported, with float constants as desirable. Algebraic notation/substitution is not required (i.e., alphabetic constants).

## 1.8 User characteristics

The arithmetic expression parser is intended for use by students, programmers, and other individuals who need to parse and evaluate arithmetic expressions.

## 1.9 Constraints

The arithmetic expression parser must be developed using the C++ programming language. The program must also be able to run on a variety of operating systems, including Linux, macOS, and Windows.

## 1.10 Assumptions and dependencies

The following assumptions are made about the arithmetic expression parser:

- The user can enter any valid or invalid arithmetic expression.
- The user may enter invalid characters.
- The user mostly uses only integer numeric constants ***(to be changed!)***

### 1.11 Requirements subsets

Requirements are sorted into two categories: **Functional Requirements** and **Supplemental Requirements**. **Functional Requirements** are programs that modify the behavior of **the program**. **Functional Requirements** can be categorized as *essential*, *desirable***,** or optional.

## 2. Specific Requirements

### 2.1 Functionality

This section and associated subsections describe the **Functional Requirements**.

#### *2.1.1 Parse arithmetic expressions*

**The program** must be able to parse and evaluate arithmetic expressions containing the following operators: +, -, *, /, %, and ^. Other requirements describe specific interactions between various arithmetic operators and operands or describe some subsystem that can be broken further away from just arithmetic expressions in general.

#### *2.1.2 Integer Numeric Constants*

**The program** supports integer numeric operands. Example: 7 * 4 or 3 / 4

#### *2.1.3 Float Numeric Constants*

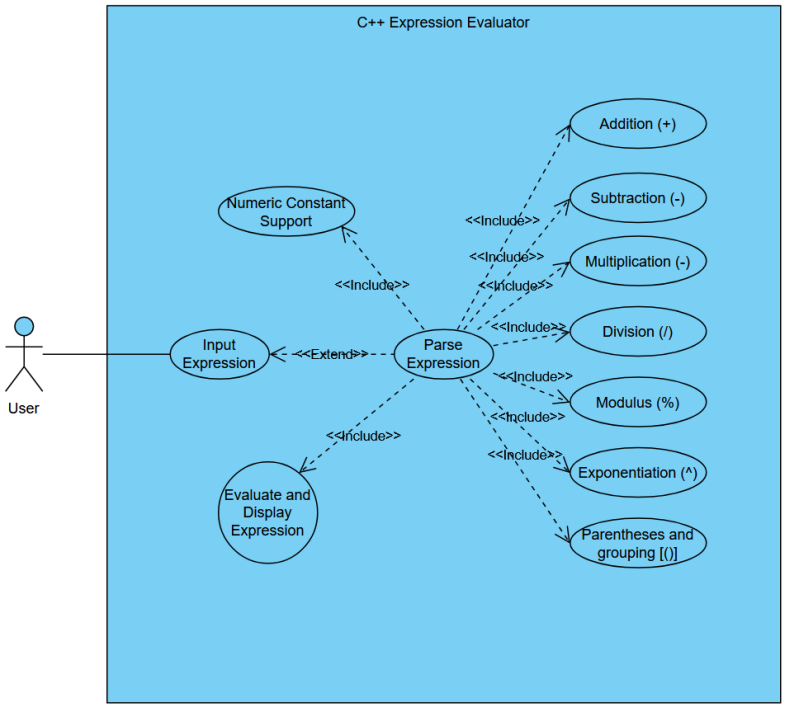
**The program** supports float numeric operands. Example: 3.14 * 2 or 3.14 / 4

#### *2.1.4 Algebraic Constants*

**The program** supports algebraic operands. Example: A * B or 1.2A. For algebraic and numeric operands containing no separator, it is assumed to a grouped expression. I.e., 1.2A/4 is equivalent to (1.2A)/4.

### 2.2 Use-Case Specifications

Below is a use-case model diagram of the requirements of **the project.** This does not model optional or desirable functional requirements.



### 2.3 Supplementary Requirements

This section and associated subsections describe the **Supplemental Requirements**.

#### *2.3.1 User README and/or Documentation*

A document or documents describing how to use **the program**.

## 3. Classification of Functional Requirements

| Functionality | Type |
| --- | --- |
| Parse/Evaluate arithmetic expressions | Essential |
| Integer Numeric Constants | Essential |
| Float Numeric Constants | Desirable |
| Algebraic Constants | Optional |

## 4. Appendices

Appendices will either be specified as part of the **SRS** or not. As of current revision, there are no appendices.