ByteMath<Company Name>

Arithmetic Expression Evaluator

User's Manual

Version 1.0

Revision History

Date	Version	Description	Author
10/12/24	1.0	Worked on sections 1, 2, 3, 4, 5, 6, 7, and 8.	Ellie Thach

Table of Contents

Test Case

1. Purpose

The purpose of the user manual is to provide users with a comprehensive guide on how to use the software for the arithmetic expression evaluator.

2. Introduction

Welcome to ByteMath's User Manual for the Arithmetic Expression Evaluator in C++. The software is designed to parse and evaluate arithmetic expressions with operators like +, -, *, /, %, and **, along with numeric constants.

- **Installation Process**
  - Download the source code from the GitHub Repository: https://github.com/aidanp12/EECS348PROJ_GROUP7/tree/main/project_artifacts
- **How to Run the Software**
  - Compile the program by using a C++ compiler (e.g. g++)
  - Execute the compiled binary file (e.g. ./a.out)

3. Getting started

- **How to Enter Expressions**
  - When prompted by the program, enter in the arithmetic expression
- **How to Use Various Operators and Functions**
  - Use parentheses and valid operators (+, -, *, /, %, and **) to ensure the correct order of evaluation
- **How to Interpret Results**
  - After entering in the arithmetic expression, the program will display the result of the evaluation

4. Advanced features

The program does not support any advanced features.

5. Troubleshooting

- **Invalid Expressions**:
  - If an invalid expression is entered, like a missing operand or an incorrect operator, the program will display an error message.
- **Division by Zero**:
  - If an attempt is made to divide an integer by zero, the program will display an error message.

6. Examples

Below are examples of how to use the software to evaluate different types of arithmetic expressions.

1. **Addition**
   a. Input: 8 + 9
   b. Output: 17
2. **Subtraction**
   a. Input: 5 – 2
   b. Output: 3
3. **Multiplication**
   a. Input: 4 * 5

   b. Output: 20
4. **Division**
   a. Input: 28 / 7
   b. Output: 4
5. **Modulus**
   a. Input: 6 % 5
   b. Output: 1
6. **Exponentiation**
   a. Input: 3**2
   b. Output: 9

7. Glossary of terms

- **Arithmetic Expression Evaluator**: A program that parses and evaluates arithmetic expressions, handling operators and numeric constants following mathematical rules.
- **Binary File**: A file whose content is in binary format.
- **C++ Compiler**: A program that translates human-readable C++ code into machine language that a computer can understand and execute.
- **GitHub Repository**: Virtual storage space for code, files, and a file's revision history.
- **Modulus**: The operator, '%', which returns the remainder of the division of one positive integer by another positive integer.
- **Numeric Constants**: Consists of numerals, optional leading sign, and an optional decimal point.
- **Parse**: Technique that is used to analyze and interpret the syntax of a text or program to extract information.

8. FAQ

Q: What is an arithmetic expression evaluator?
A: An arithmetic expression evaluator calculates the result of an arithmetic expression.

Q: Would this program be able to be implemented into a larger compiler project?
A: Yes. The software is designed to be able to be implemented into a larger compiler project for a language L that is being developed in C++.

Q: How to handle complex expressions with nested parentheses?
A: The program is designed to recognize and evaluate expressions inside parentheses according to the order of operations.

Q: Why are parentheses important in arithmetic expressions?
A: Parentheses are important in arithmetic expressions because they can override the usual order of operations. For example, in the expression '3 * (5+2)', the addition is done first because it is inside the parentheses even though addition comes after multiplication in the order of operations.

Q: What happens if a parenthesis is not closed in an arithmetic expression?
A: The program will detect that a parenthesis does not have a match and output an error message.

Q: What happens if an invalid character is in an arithmetic expression?
A: The program will detect that an invalid character is in the expression and output an error message.