348 Industries

EECS 348 Fall 2023 Team Project: Arithmetic Evaluation Program

User's Manual

Version <1.0>

Revision History

Date	Version	Description	Author
12/01/23	1.0	This serves as the User Manual for the Arithmetic Evaluation Program. It is the guidelines for using the program, and how to preserve and maintain it. This was exclusively developed for the EECS 348 Fall 2023 Team Project.	Dev Patel Jay Chung Ruth Higgason Colin Marett

Table of Contents

Test Case

1. Purpose

The purpose of this user manual is to serve as a comprehensive guide for users, enabling them to navigate and utilize the software efficiently. It aims to provide clear instructions and explanations on various aspects of the software, ensuring that users can make the most out of its functionality. The user manual also helps users troubleshoot common issues and understand the benefits and advantages of using the software.

2. Introduction

The software user manual provides users with an easy-to-understand guide on how to use the software effectively. This section offers a brief overview of the software, including its purpose, key features, and instructions on how to install and run it. By reading this section, users will gain a better understanding of the software and its capabilities.

3. Getting started

The "Getting Started" section serves as a step-by-step guide for users to begin using the software and evaluate arithmetic expressions. It includes instructions on how to enter expressions, utilize the different operators and functions provided by the software, and interpret the results accurately. By following the instructions in this section, users will be able to quickly grasp the fundamentals of using the software and perform arithmetic evaluations effectively.

4. Advanced features

4.1 Handling of Complex Expressions

- Nested Parentheses: The software adeptly manages complex expressions with nested parentheses, ensuring the correct order of operations is applied.
- Unary Operators: Supports the use of unary plus and minus operators, allowing for expressions like `-(+2) * (+3)`.

4.2 Support for Various Operators

- Arithmetic Operators: Includes `+`, `-`, `*`, `/`, `%`, and `^` for a wide range of arithmetic operations.
- Exponentiation: The software recognizes `^` for exponentiation, enabling power calculations.

4.3 Error Handling

- Error Detection: Detects and handles errors such as division by zero or invalid input format.
- Input Validation: Ensures only numeric expressions and supported operators are entered.

5. Troubleshooting

5.1 Common Problems and Solutions

1. Unmatched Parentheses
   - Problem: If parentheses are not properly matched, the program will not evaluate the expression.
   - Solution: Ensure every opening parenthesis has a corresponding closing parenthesis.

2. Division by Zero
   - Problem: Attempting to divide a number by zero.
   - Solution: Modify the expression to avoid division by zero.

3. Invalid Operator Usage
   - Problem: Using unsupported characters or operators.
   - Solution: Use only the supported operators: `+`, `-`, `*`, `/`, `%`, `^`.

4. Incorrectly Formatted Input
   - Problem: Entering expressions with missing operands or operators.
   - Solution: Ensure all operators have the required operands.

5. Handling Negative Numbers
   - Problem: Issues with inputting negative numbers.
   - Solution: Enclose negative numbers in parentheses (e.g., `(-3)`).

6. Software Not Responding to Input
   - Problem: The program might not respond due to invalid or empty input.
   - Solution: Check for empty strings or invalid characters. Ensure the expression follows the correct arithmetic format.

7. Unexpected Results
   - Problem: Receiving results that do not match expectations.
   - Solution: Recheck the expression for operator precedence and parenthesis usage.

6. Examples

This section provides various examples to demonstrate how to use the Arithmetic Expression Evaluator software.

6.1 Basic Operations

1. Addition
   - Input: `3 + 4`
   - Result: `7`

2. Subtraction
   - Input: `8 - 5`
   - Result: `3`

3. Multiplication
   - Input: `6 * 7`
   - Result: `42`

4. Division
   - Input: `20 / 4`
   - Result: `5`

5. Modulo
   - Input: `9 % 4`
   - Result: `1`

6. Exponentiation
   - Input: `2 ^ 3`
   - Result: `8`

6.2 Complex Expressions

1. Mixed Operations
   - Input: `4 + 3 * 2 - 1`
   - Result: `9`

2. Using Parentheses
   - Input: `(3 + 2) * (7 - 4)`
   - Result: `15`

3. Nested Parentheses
   - Input: `((2 + 3) * (4 - 2)) ^ 2`
   - Result: `100`

4. Combined Division and Multiplication
   - Input: `10 * 2 / 5`
   - Result: `4`

5. Complex Mixed Operators
   - Input: `4 * (3 + 2) % 7 - 1`
   - Result: `5`

6.3 Handling Negative Numbers and Unary Operators

1. Negative Numbers
   - Input: `(-3) * 4`
   - Result: `-12`

2. Unary Negation
   - Input: `-(2 + 3)`
   - Result: `-5`

3. Combination of Unary and Arithmetic Operators
   - Input: `-(+2) * (+3) - (-4) / (-5)`
   - Result: `-6.8`

6.4 Error Examples

1. Division by Zero
   - Input: `5 / 0`
   - Expected Behavior: Error message indicating division by zero.

2. Invalid Expression

- Input: `2 + * 3`
- Expected Behavior: Error message indicating an invalid expression.

7. Glossary of terms

- Arithmetic Expression: A combination of numbers, operators, and sometimes parentheses that define a particular calculation order.
- Operand: A quantity on which an operation is performed, typically a number in an arithmetic expression.
- Operator: A symbol representing a mathematical operation such as addition (+), subtraction (-), multiplication (*), division (/), modulus (%), or exponentiation (^).
- Unary Operator: An operator that takes only one operand. In the context of this software, unary plus (+) and unary minus (-) are examples.
- Binary Operator: An operator that takes two operands and performs a calculation. For instance, `+` in `3 + 4`.
- Parentheses: Symbols used in arithmetic expressions to alter the standard order of operations. Operations enclosed in parentheses are performed first.
- Nested Parentheses: Parentheses within parentheses that indicate operations to be performed in the innermost set first.
- Modulo: An operation that finds the remainder after division of one number by another, denoted by `%`.
- Exponentiation: An operation that raises a number to the power of another number, indicated by `^`.
- Syntax Error: An error that occurs when the structure of an arithmetic expression is incorrect (e.g., unmatched parentheses or missing operands).
- Runtime Error: An error that occurs while the program is running, such as division by zero.
- CLI: Command Line Interface, which means users interact with the software by typing commands into a console or terminal.
- Parsing: The process of analyzing a string of symbols, either in natural language or computer languages.
- Tokenization: The process of converting a sequence of characters into a sequence of tokens, which can be used for further processing like parsing.
- Integer: A whole number that can be positive, negative, or zero but has no fractional part.
- Floating Point Number: A number that has a decimal place, allowing for fractional values.

8. FAQ

What is the Arithmetic Problem Solver Program?

The Arithmetic Problem Solver Program is a C++ application designed to assist users in solving various arithmetic problems, including basic calculations, equations, and mathematical expressions.

How do I input a mathematical expression into the program?

You can input mathematical expressions directly into the program by using the standard input methods in C++. Simply enter the expression as you would on paper, using appropriate operators (+, -, *, /), and press Enter to get the result.

What types of arithmetic problems can the program solve?

The program can handle a wide range of arithmetic problems, including addition, subtraction, multiplication, division, and solving simple equations. It supports both integers and floating-point numbers.

Can I solve complex equations with the program?

While the program is primarily designed for basic arithmetic operations, it can handle simple linear equations. However, for more complex algebraic or trigonometric equations, it is recommended to use specialized software or libraries.

How do I handle errors or invalid inputs?

The program includes error-checking mechanisms to handle invalid inputs. If you enter an expression with syntax errors or attempt an operation that is not supported, the program will provide an error message to guide you.

Is there a limit to the size of the mathematical expressions I can input?

The program is designed to handle reasonably large mathematical expressions. However, extremely long or complex expressions may exceed the program's capacity. In such cases, consider breaking down the expression into smaller parts and solving them separately.

Is there a user interface (UI) for the program?

The Arithmetic Problem Solver Program primarily operates through the console or command line interface. It does not have a graphical user interface (GUI). If you need a GUI, you may want to create a front-end application that interfaces with the program's functions.

Where can I find additional help or support?

For further assistance, refer to the program's documentation and comments within the source code. Additionally, you can seek help from online forums, community discussions, or the developer community associated with the C++ programming language.