Taus

Arithmetic Expression Evaluator

Test Case

Version 1.0

Revision History

Date	Version	Description	Author
11/12/24	1.0	Began considering the possible corner cases and completing documentation.	Charlie Doherty, Jonathan Gott, Will Calhoun, Josh Dwoskin, Calder Kamman, Caleb Hite

Table of Contents

1. Purpose 4

2. Test case identifier 4

3. Test item 4

4. Input specifications 4

5. Output specifications 4

6. Environmental needs 4
   - 6.1.1 Hardware 4
   - 6.1.2 Software 4
   - 6.1.3 Other 4

7. Special procedural requirements 5

8. Intercase dependencies 5

Test Case

1. Purpose

This Test Case Specification document for the Arithmetic Expression Evaluator defines a comprehensive set of test cases to ensure the accurate and reliable evaluation of various arithmetic expressions. The primary objectives of these tests are to verify correctness of the program and find corner cases that are normally overlooked.

2. Test case identifier

TC001 - Basic Arithmetic Operations
TC002 - Operator Precedence
TC003 - Parentheses Handling
TC004 - Division and Modulo by Zero
TC005 - Exponentiation Operation
TC006 - Handling Whitespace
TC007 - Complex Expressions

3. Test item

TC001 - Ensure that the basic arithmetic operations (addition, subtraction, multiplication, division, and modulo) are correctly implemented and return the expected results.

TC002 - Verify that the program correctly handles operator precedence, ensuring that operations with higher precedence are executed before those with lower precedence.

TC003 - Confirm that the program correctly evaluates expressions with parentheses, ensuring that operations within parentheses are executed first.

TC004 - Ensure that the program correctly handles division and modulo by zero, throwing the appropriate runtime error.

TC005 - Verify that the exponentiation operation is correctly implemented and returns the expected results.

TC006 - Confirm that the program correctly handles and ignores whitespace within the expression.

TC007 - Ensure that the program can correctly evaluate complex expressions that combine multiple operations, parentheses, and different operator precedences.

4. Input specifications

The input should be formatted as a text string that represents an arithmetic expression. It may contain “() + - * / ^ %” as well as any digits from 0-9 in between. External characters will be handled during runtime. This is true for all test cases. Example: (9+1)*51/2.

5. Output specifications

Given valid terminal input, the output will be a floating point number in the terminal as well. Example: 10.45. Given invalid input, the output will be a prompt asking the user to re-enter an expression. This is true for all test cases.

6. Environmental needs

A computer capable of downloading and running C++ programs and a C++ compiler. Visual Studio or Visual Studio Code is recommended for best terminal I/O. This is true for all test cases.

7. Special procedural requirements

The test cases should be executed in a controlled environment, use every operation, and use various combinations of integers and floating point numbers. This is true for all test cases.

8. Intercase dependencies

All test cases are independent of other test cases.