Group 32

Arithmetic Expression Evaluator in C++

User's Manual

Version 1.1

Revision History

Date	Version	Description	Author
01/12/2023	1.0	Filled in various empty areas with information.	Manu R.
03/13/2023	1.1	Quality check, revisions, and final touches.	Jordan B.

Table of Contents

Test Case

1. Purpose

The user’s manual for the Arithmetic Expression Evaluator in C++ is a resource provided to educate users on the capabilities and usage of the software.

2. Introduction

Welcome to the Arithmetic Expression Evaluator, a program designed to take arithmetic expressions as input, evaluate them, and print the result. This manual provides detailed information on how to use the program effectively.

3. Getting started

This section should provide a step-by-step guide on how to use the software to evaluate arithmetic expressions. It should include instructions on how to enter expressions, how to use the various operators and functions, and how to interpret the results.

3.1 Supported Platforms

The Arithmetic Expression Evaluator is written in C++ and is compatible with platforms that support C++ development.

3.2 Software Requirements

Ensure you have the following software installed on your system:

- C++ Compiler: A compatible C++ compiler to build and run the program.

3.3 Installation

No installation is required for this program. To access the program, follow the steps below:

3.3.1 Accessing the Code

1. Copy the provided C++ code. (Titled *finalproject_v1.9*).
2. Open a C++ development environment or text editor.
3. Paste the code into a new or existing file.

3.3.2 Compiling and Running

1. Compile the code using your C++ compiler.
2. Run the compiled executable.

4. Usage

Upon running, the program will prompt you to enter an arithmetic expression. After inputting the expression, the program will evaluate it and print the result.

4.1 Operators

The user has access to the following operators:

- + (Addition operator)
  Inputting this operator between two values or expressions returns their sum.
  Example: An input of “2+2” would return 4.

- - (Subtraction operator)

Inputting this operator between two values or two expressions returns their difference.

Example: An input of “2-2” would return 0.

- * (Multiplication operator)

  Inputting this operator between two values or two expressions returns their product.

  Example: An input of “100*4” would return 400.

- / (Division operator)

  Inputting this operator between two values or two expressions returns the quotient.

  Example: An input of “16/8” would return 2.

- % (Modulo operator)

  Inputting this operator between two values or two expressions returns the remainder.

  Example: An input of “17%2” would return 1.

- ^ (Exponent operator)

  Inputting this operator and a number after a value or an expression returns the value raised to that power.

  Example: An input of “5^2” would return 25.

5. Advanced features

The software has the following advanced features.

- **Expression Parsing + PEMDAS**
  The program is capable of parsing the user's input and utilizing operator precedence (PEMDAS.)
- **Parentheses Usage**
  The program can additionally handle parentheses, allowing for a custom parsing order and multiple operations to be completed in one input.

6. Troubleshooting

6.1 Invalid Expression: Missing Parenthesis

If the program encounters an imbalance in the number of opening and closing parentheses, it will display an error message. To avoid this, be sure to match all parentheses.

Example:

```
Enter expression: 2 * (3 + 4
Invalid expression: Missing parenthesis.
```

6.2 Invalid Expression: Division by Zero

An attempt to divide by zero will result in an error message, and the program will terminate.

Example:

```
Enter expression: 8 / 0
```

```
Invalid expression: Division by zero.
```

6.3 Invalid Expression: Invalid Operator Sequence

If an invalid operator sequence is detected within parentheses, the program will display an appropriate error message.

Example:

```
Enter expression: (2+3*) 5
Invalid expression: Invalid operator sequence.
```

6.4 Invalid Expression: Invalid Character

If an invalid character is entered in the expression, the program will terminate and display an error message. To avoid this, only input integer numbers and acceptable operators (listed in 4.1).

Example:

```
Enter expression: aaa
Invalid expression: Invalid character.
```

7. Acknowledgements

This program is based on the work of Derek Norton, Ryan Grimsley, Jordan Burns, Manu Redd, and Priyatam Nuney.