Arithmetic Expression Evaluator

User’s Manual

Version 1.5

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 12/02/2023 | 1.0 | Purpose and Introduction | Elizabeth Channel |
| 12/02/2023 | 1.1 | FAQ | C. Cooper |
| 12/02/2023 | 1.2 | Getting started & Advanced Features | Aryan Kevat |
| 12/03/2023 | 1.3 | Add Troubleshooting Section | Vivian Lara |
| 12/03/2023 | 1.4 | Add Glossary of Terms | Jeff Burns |
| 12/03/2023 | 1.5 | Add Examples | Ashley Vierling |

Table of Contents

[1. Purpose 3](#_Toc152528200)

[2. Introduction 3](#_Toc152528201)

[3. Getting started 3](#_Toc152528202)

[4. Advanced features 4](#_Toc152528203)

[5. Troubleshooting 4](#_Toc152528204)

[6. Examples 4](#_Toc152528205)

[7. Glossary of terms 5](#_Toc152528206)

[8. FAQ 5](#_Toc152528207)

[8.1 Can you input floating point values? 5](#_Toc152528208)

[8.2 How do you exit the program? 5](#_Toc152528209)

[8.3 What is the range of values that you can input? 6](#_Toc152528210)

[8.4 What operations are supported? 6](#_Toc152528211)

[8.5 Does the program support implied multiplication with parentheses? 6](#_Toc152528212)

[8.6 What order of operations does the program use? 6](#_Toc152528213)

[8.7 Where can I find the program’s source code? 6](#_Toc152528214)

Test Case

# Purpose

The purpose of this document is to act as a reference for use of the *Arithmetic Expression Evaluator*. Users of the program can look to this for information on how to use the evaluator. It will include the following sections: Introduction, Getting Started, Advanced Feature, Troubleshooting, Examples, Glossary of Terms, and FAQs.

# Introduction

This software is an arithmetic expression evaluator. It is a program that takes an input of an arithmetic expression and calculates the results in observance of PEMDAS. Features of this program include an easy-to-use input command line. It supports the following operators: Addition (+), Subtraction (-), Multiplication (\*), Division (/), Modulo (%), and Exponentiation (^). To install the program simply download the source folder. To run the evaluator, compile and run the program as required by your specific system.

# Getting started

**Installation:** In order to install the *Arithmetic Expression Evaluator*, use a web browser to navigate to the [GitHub repository for the project](https://github.com/FlyingTNT/EECS-348-Project). In the base directory, there will be available binaries. calculator-win.exe corresponds to the Windows executable and calculator-linux corresponds to the Linux native binary. Download the file corresponding with your operating system.

**Running:** To run the program, open a terminal or console of choice, and navigate to the directory where you downloaded the binary. Run the file by typing ./ followed by the name of the file, then press enter. You will see a prompt for user input.

**Usage:** You may input an arithemetic expression containing the digits 0-9, symbols +, -, \*, /, %, ^, (, or ), and the space character. Each character operates the same way as their mathematical counterparts, with ^ representing the exponent operation, and % representing the modulus operation. In addition, the + and - symbols can be used to express absolute value and negation respectively when placed in front of another numeric value or expression enclosed within parentheses. While decimal input is not explicitly supported, you can achieve similar results by using the fraction equivalent. For example, 0.5 can be expressed as (1/2). Please note than you cannot correct any errors in the input after they are typed. When you are finished entering an expression or wish to clear the input, press enter for the software to evaluate it. If you wish to exit the program, type “exit”.

**Results:** If given a valid expression as input, the software will output the equivalent numeric value of the expression. The expression is evaluated following PEMDAS order of operations. If given as “exit” as input, the system will exit and return you to the command line. If the input is not valid, the software will output an error message. The error message can either indicate a problem with the structure of the input (missing operands, umatched parentheses, missing operators, etc) or a problem with the mathematical evaluation (division by zero, modulus with a negative divisor, etc).

# Advanced features

There are not any advanced features within the software. All features are expressed as a combination of the numeric digits and operators defined in the usage documentation.

# Troubleshooting

|  |  |
| --- | --- |
| Common Problems | Solutions |
| Error:\_ is missing an operand | Ensure each operator is accompanied by the appropriate operands. Check for missing numbers or operators in the expression. |
| Error: cannot divide/modulo by zero | These operations will result in undefined and therefore are not possible. |
| Error: illegal symbol: \_ | Ensure you are using the correct corresponding symbols. Use ^ for exponentiation (not \*\*) and / for division (not //). |
| Error: missing operator! | Ensure you do not have extra numbers or are missing an operator before/after a number. |
| Error: unmatched parentheses | Ensure all parentheses are a pair of opened and closed. |
| Error: input overflow | Input exceeded the maximum value of int. Please try again with a value less than 2147483647. |
| Cannot exit calculator | To exit, simply type exit . |

# Examples

To use this software the expression must be put into the comand line. Once the expression has been written in the command line the user must hit enter to submit the expression for evaluation. Examples of simple arithmetic are:

1. Command line input: 3+4+6+2
   1. Result: 15
2. Command line input: 4-5+7
   1. Result: 6
3. Command line input: (8+9)-(6%4)
   1. Result: 15
4. Command line input: 2^(5+3)
   1. Result: 256
5. Command line input: 3^2^3
   1. Result: 6561
6. Command line input: 14--(23^2 % 7 +8) \* 174^+-0
   1. Result: 26
7. Command line input: (5/2) \* ((6/5) + (33/10)) - (1/2)
   1. Result: 10.75
8. Command line input: 8+4-(2\*2)+3^2
   1. Result: 17
9. Command line input: 38-25-9
   1. Result: 4
10. Command line input: 21/5%3\*17
    1. Result: 17

Examples of errors:

1. Command line input: 5\*\*7
   1. Result: Error: \* is missing an operand!
2. Command line input: 7-5 4\*3
   1. Result: Error: missing operator!
3. Command line input: (2@3)/2
   1. Result: Error: illegal symbol: @!
4. Command line input: 5/0
   1. Result: Error: cannot divide by zero!
5. Command line input: 5%0
   1. Result: Error: cannot modulo by zero!
6. Command line input: ((3-5)\*-2-(4+5)))-((3%7)
   1. Result: Error: unmatched parentheses!
7. Command line input: ((17+5)\*23%7
   1. Result: Error: unmatched parentheses!
8. Command line input: (7%2)%(2%1)
   1. Result: Error: cannot modulo by zero!
9. Command line input: 4^
   1. Result: Error: ^ is missing an operand

# Glossary of terms

Modulo/modulus: Operation which produces the remainder of an integer division.

Operand: An object that is operated on by an operator.

Operator: A symbol that represents an operation to be performed on one or more operands.

PEMDAS: acronym used to show the order of operations in arithmetic expressions. It stands for **P**arentheses, **E**xponents, **M**ultiplication, **D**ivision, **A**ddition, and **S**ubtraction.

# FAQ

## Can you input floating point values?

No; direct input of floating-point values is not supported. However, you can use floating point values by representing them fractionally, such as using (3/2) to represent 1.5.

## How do you exit the program?

By inputting “exit” or closing the window.

## What is the range of values that you can input?

You can input values with a magnitude less than or equal to (2^31)-1. This is because that is the maximum value a 32-bit signed integer can store.

## What operations are supported?

The supported operations are addition, subtraction, multiplication, division, modulo, exponentiation, negation, and absolute value. Parentheses are supported for grouping. Division is always floating-point.

## Does the program support implied multiplication with parentheses?

No; the program does not support implied multiplication.

## What order of operations does the program use?

There are five levels of precedence; these are (in the order they are evaluated):

1: Parentheses

2: Exponentiation

3: Multiplication, Division, and Modulo

4: Absolute Value and Negation

5: Addition and Subtraction

Within each level, operations are evaluated left-to-right.

## Where can I find the program’s source code?

The source code can be found at [github.com/FlyingTNT/EECS-348-Project](https://github.com/FlyingTNT/EECS-348-Project).