Arithmetic Expression Evaluator

User Manual

Version 1.0.1

Revision History

Date	Version	Description	Author
11/29/2023	1.0	Document creation and initial details added	Charlie Gillund
12/2/2023	1.0.1	Filled out extra details and revamped document.	Jake Bernard

Table of Contents

User Manual

1. Introduction

The Arithmetic Expression Evaluator is a command line program able to evaluate and display the results of arithmetic expressions entered by the user. Some extra quality-of-life functionality and features are also present, such as in-program help and a history of calculated expressions.

Supported mathematical operations are:

- Addition (+)
- Subtraction and negation (-)
- Multiplication (*)
- Division (/)
- Exponentiation (^ or **)
- Modulo (%)
- Grouping expressions within parentheses

Supported mathematical constants are:

- π (pi)
- Euler's number (e)

2. Getting started

2.1 Setup

First, clone or download the GitHub repo containing the project (link). If you've downloaded a compressed version of the project, be sure to extract the "full_program" folder from the archive.

Next, navigate into the "full_program" folder and find the folder matching your operating system, with the current choices being "Windows" or "Linux".

Finally, either launch run.exe from the command line or as a standalone application (on Windows), or use the command line to launch run (on Linux).

2.2 Using the program

Upon startup, the user will be greeted with a welcome message and a prompt in the main menu to enter one of the program's keywords which access the different areas of functionality, or to type the "help" keyword. These keywords, along with all other input, is case-sensitive.

2.3 Main Menu

The keywords a user can enter in the main menu, along with their associated functionality, are:

- **help**

  Displays all the keywords in the program along with a brief description of each. Immediately returns to the main menu.

- **eval**

  Enters the user into evaluation mode, in which they can enter an arithmetic expression using the built-in operators and numeric constants. See *2.4 Evaluation Menu* for more information.

  Alternatively, can be used to immediately evaluate an expression from the main menu by following the keyword with the desired expression to be evaluated.

- **history**

  Displays a history of all calculated expressions and allows the user to select an expression to view the result again.

  See *2.5 History Menu* for more information.

- **exit**

  Exits the program.

2.4 Evaluation Menu

In the evaluation menu, users can input arithmetic expressions using built-in supported operations and numeric constants and have the result returned to them

After pressing enter, the result will be displayed. Incorrectly formatted or mathematically invalid expressions will result in an error being displayed.

For all supported operations and constants, see *2.6 Supported Operations and Constants.*

2.5 History Menu

The history menu allows the viewer to see the last eight (8) expressions which have previously been entered and evaluated, successfully or not.

By entering the number of an expression the user would like to see the result of again, the program will then display both the expression and its result to the user and then return them to the main menu.

Any other input returns the user to the main menu.

2.6 Supported Operations and Constants

Supported mathematical operations are:

- Addition (+)
- Subtraction and negation (-)
- Multiplication (*)
- Division (/)
- Exponentiation (^ or **)
- Modulo (%)
- Grouping expressions within parentheses

Supported mathematical constants are:

- π (pi)
- Euler's number (e)

3. Advanced features

For more examples of each, see *Section 5, Examples*.

3.1 Implicit Multiplication

When evaluating an expression, users may multiple or constants within groups of parentheses by placing them adjacent to others. A group of parentheses must be separating one of the expressions from the other. For instance, the expression "9 (1 + 1)" would be expanded to "9 * (1 + 1)".

3.2 Unary negation

A minus sign can also be used as a unary operator that negates the number to its right. This is triggered automatically when minus signs are placed in parts of the expression that binary operators cannot be placed normally, such as at the beginning of an expression or after another operator.

3.3 Multiple exponentiation operators

Exponentiation can use either the ^ operator or can use two asterisks in a row **.

4. Troubleshooting

Some errors messages that might be encountered while running the program and their meanings are detailed below. It should be noted in the program that indexing starts at 0 for all expressions.

4.1 Unknown word

Message
Error: Unknown word "<example word>" in expression
Meaning
The user has entered a word not in the program's vocabulary which cannot be recognized.

4.2 Mismatched parentheses

Message
Error: Mismatched parentheses at position x
Meaning
The user has input a right parenthesis character in an invalid position with no matching left parenthesis.

4.3 Unusable Character

Message
Error: Unusable character 'x' at position y
Meaning
The user has entered a character that is not in the program's vocabulary at the given position in the expression.

4.4 Number follows number

Message
Number follows number at position x
Meaning
The user has entered an expression in which two numbers (separated by a whitespace) follow each other.

4.5 Operator follows operator

Message
Error: Operator follows operator at position x
Meaning
The user has entered an expression in which two binary operators follow each other.

4.6 Binary operator incorrectly placed

Message
Error: Binary operator 'x' incorrectly placed at position y
Meaning
The user has entered an expression in which a binary operator is placed in an invalid position, such after a left parenthesis or at the very beginning of the expression.

4.7 Empty Parentheses

Message
Error: Empty parentheses at position x
Meaning
The user has input a pair of parentheses which encloses nothing.

4.8 Operator precedes right parenthesis

Message
Error: Operator precedes right parenthesis at position x
Meaning
The user has input an operator directly preceding a closing parentheses.

4.9 Unbalanced parentheses

Message

Error: Parentheses in expression are not balanced
Meaning
The amount of opening and closing parentheses in the expression are not equal.

4.10 Operator at end of expression

Message
Error: Operator at the end of expression
Meaning
An operator is at the end of the expression and cannot be evaluated.

4.11 Expression generates NaN

Message
Error: Expression generated NaN during evaluation
Meaning
The expression could not be evaluated because an operation within the expression produced a NaN floating point value. This can happen when taking negative square roots, for instance.

4.12 Exceeds floating point size

Message
Error: Exceeds floating point size limit (generated infinity during evaluation)
Meaning
A value generated by evaluating the expression exceeded the size that floating point numbers can accomodate and an "inf" value was generated.

4.13 Division by zero

Message
Error: Division by zero encountered in expression
Meaning
Somewhere in the expression a division by zero occurs, so the expression cannot be evaluated.

4.14 Modulo by zero

Message
Error: Modulo by zero encountered in expression
Meaning
Somewhere in the expression a modulo by zero occurs, so the expression cannot be evaluated.

5. Examples

The following are examples of user interaction with the program.

The format in which the following examples are presented is demonstrated below:

This is a message from the program.

This is still a message from the program.

>this is user input

5.1 Accessing help

```
Welcome to the Arithmetic Expression Evaluator!
Enter a command below, type "help" for help:
>help
Supported operators:
+, -, /, *, %, ^ (or **)
Current commands are:
- "eval": allows you to enter an expression to be evaluated
        - alternvatively, enter "eval <your expression>" (without angle brackets)
- "exit": exits program
- "history": displays history and allows selecting a previous expression
- "help": displays this stuff
```

5.1 Accessing help

```
Welcome to the Arithmetic Expression Evaluator!
Enter a command below, type "help" for help:
>help
Supported operators:
+, -, /, *, %, ^ (or **)
Supported numerical constants:
pi, e
Current commands are:
- "eval": allows you to enter an expression to be evaluated
        - alternvatively, enter "eval <your expression>" (without angle brackets)
- "exit": exits program
- "history": displays history and allows selecting a previous expression
- "help": displays this stuff
```

5.3 Entering a basic expression

```
Enter a command below, type "help" for help:
>eval
Enter expression to be evaluated or type "exit" to exit.
>1 + 2
Result:
3.00
```

5.4 Entering a longer expression

```
Enter a command below, type "help" for help:
>eval
Enter expression to be evaluated or type "exit" to exit.
>88 * 9 - 5 % 4 / 2
Result:
791.50
```

5.5 Entering an expression with exponentiation in two ways

```
Enter a command below, type "help" for help:
>eval
Enter expression to be evaluated or type "exit" to exit.
>2 ^ 3
Result:
8.00

Enter a command below, type "help" for help:
>eval
Enter expression to be evaluated or type "exit" to exit.
>2 ** 3
Result:
8.00
```

5.6 Using immediate evaluation

```
Enter a command below, type "help" for help:
> eval 7 - 1 + 0
Result:
6.00
```

5.7 Entering an expression with parenthetical grouping

```
Enter a command below, type "help" for help:
>eval
Enter expression to be evaluated or type "exit" to exit.
>(1 + 1) * (2 - (3 * (1 / 3)))
Result:
2.00
```

5.8 Using implicit multiplication

```
Enter a command below, type "help" for help:
>eval
Enter expression to be evaluated or type "exit" to exit.
>1 + 2(3 - 1)
Result:
5.00
```

5.9 Using unary negation

```
Enter a command below, type "help" for help:
>eval
Enter expression to be evaluated or type "exit" to exit.
>----3
Result:
3.00
```

5.10 Accessing history

```
Enter a command below, type "help" for help:
>history
Current history:
      1) 444 ** 2
      2) 1 % 6 * 2
      3) 1 + 3 - 0
      4) ----3
Choose an option with corresponding number, or enter anything else to return to the main menu.
>1
Expression:
444 ** 2
Result:
197136.00
```

6. Glossary of terms

6.1 Expression

A combination of numbers and mathematical operators which can be evaluates to a single value when the operations of the operators are performed upon the numbers within the expression.

6.2 Operator

A mathematical construct which can, in some way, modify a number or numbers.

6.3 Modulo

A mathematical operator, represented here by %, that returns the remainder of a division operation between two numbers. For example:
7 % 4 = 3, as the result of 7 / 4 is 1 remainder 3.

6.4 Unary/Binary

When referring to mathematical operators, references the amount of arguments that are taken.

A **unary** operator, such as when the - symbol is used for negation, takes only one argument.
For example, the only argument to the leftmost - in this expression is -3:
-(-3) = 3

A **binary operator** takes two arguments -- usually one on the left and one on the right.
For example, the + here takes 2 and 3 as arguments:
2 + 3 = 5

7. FAQ

7.1 Are decimal points accepted in input?

No, as this was not in the requirements for this program.

7.2 Why are some of my answers not exact?

This is a limitation of floating point numbers, which is what this program uses to store numerical values. If you would like to learn more about floating point numbers, plenty of great computer science courses are offered at the University of Kansas. Alternatively, you can find a wealth of knowledge about computer science topics online, but you won't be able to get a degree and you might miss out on networking opportunities.

7.3 How does modulo work with floating point numbers?

The numbers are cast first to integers (not rounded) and then the modulo operation is performed on integer values of the numbers.