University of Kansas EECS 348

Arithmetic Expression Evaluator in C++
User's Manual

Version <1.5>

Revision History

Date	Version	Description	Author
12/05/24	1.0	Adding to Purpose	Max
12/10/24	1.1	Finished Purpose section and started Getting Started section.	Max
12/10/24	1.2	Filled out introduction, FAQ, glossary, troubleshooting.	Eliza
12/10/24	1.3	Added more QA to FAQ	Daniel
12/10/24	1.4	Filled out example section	Tyler
12/11/24	1.5	Finished Getting Started and Advanced Features sections.	Max

Table of Contents

Test Case

1. Purpose

The purpose of the User's Manual document is to provide any potential users with a full, in-depth guide to the Arithmetic Expression Evaluator (AEE). The document includes an introduction to everything about the AEE, including its purpose, features, how to install, and how to use the program.

Additionally, the document also provides examples of the software's output for different types of arithmetic expressions the user could put in, examples of problems and their solutions, and a list of any technical terms used in the manual.

2. Introduction

The Arithmetic Expression Evaluator (AEE) takes in a mathematical expression using symbols: 0-9, +, -, *, /, %. After this expression is taken in, it is run through a process to either evaluate it or conclude that the expression is mathematically impossible. Since this project is meant to be included in a larger system, it has a very limited interface to ensure less work and parsing for this larger system.
The AEE features a place of input, and a line of output with the result, error, or quit message. This is to ensure conscience and simplicity for the case that the AEE is included in a larger system. If this had not been the case, the team would have done a more elaborate user interface, but unfortunately, because of the limitation of being included in a larger system that was not favorable.
To install and use the AEE, either download the exe or all the files from the Github repository, putting the file(s) all in the same folder or directory. Finally, head to a command terminal on the computer that the AEE is installed on (preferable KU Linux servers or Linux command terminal) and either compile the makefile using command "make" or directly run the application using appropriate commands for the terminal (for KU Linux use "make" and then "./implementation").
Once you have started the AEE program enter whatever expression you wish and have the answer instantly.

3. Getting started

Upon starting the Arithmetic Expression Evaluator (AEE), the user will be prompted to input an arithmetic equation for the program to evaluate. As long as it is valid, the user is able to enter an expression, such as "10*2/5", to have the AEE breakdown using PEMDAS to calculate a final answer.
The user can enter any number of expressions one at a time, as long as they follow the guidelines for a valid input. The only characters allowed by the user are the following:

- Numerical Values: Whole integers composed of 0-9. For example, "54", "3", or "123" would all be valid inputs within an equation.
- + : Used between two integers to find their sum. For example, "3+7" would be a valid input, and would result in "10".
- - : Used between two integers to find their difference. For example, "7-3" would be a valid input, and would result in "4".
    - This can also be used to signify a negative integer. For example, "-3", "-79", and "-245" would all be valid inputs within an equation.
- * : Used between two integers to find their product. For example, "9*8" would be a valid input, and would result in "72".
- / : Used between two integers to find their quotient. For example, "10/2" would be a valid input, and would result in "5".
- %: Used to between two integers to find their modulus. For example, "15%4" would be a valid input, and would result in "3".
- ** : Used between two integers to find what the first number is to the power of the second. For example, "5**2" would be a valid input, and would result in "25".

- Parenthesis: Both parentheses brackets, "(" and ")" can be used to separate out equations, as long as the number of each bracket remains the same. For example, "(5+7)*2" would be a valid input, and would result in "24".

Each of these characters used in their correct placement in an equation will allow the AEE to break it down and give the user an output. If the AEE detects any character not listed above, or finds that an equation is not formatted correctly, such like "*5+2", which lacks a numerical value next to "*", or there is a decimal input, it will output an invalid input error.

4. Advanced features

Exponentiation of Negatives - The AEE is capable of recognizing and solving negative exponential numbers. Normally, when a number is exponentiated by a positive integer, the number will increase, however, when exponentiated by a negative integer, it will decrease. For example, if the user were to input "5**(-3)", the AEE would be able to recognize the answer to be "0.008".

5. Troubleshooting

If an error message is received, read the statement after the error and find how it relates to the expression inputted. For example, if the message is "error: invalid input of operators" there may be improper placement of an operator, or a missing operator.

If there is a segmentation error or core dump, this is an error within how the Arithmetic Expression Evaluator handles the numeric input, and most likely the result or one of the numbers in the input was too large for the computer to handle. If this happens, restart the program and avoid entering extremely large numbers or numbers outside the range of 1.8×10^{308}.

6. Examples

Addition and Subtraction

- Addition and subtraction use the "+" and "-" operators respectively.
  - Example expression 2+3-5

Exponentiation

- Correct exponent format: 2**3
- The AEE will only consider expressions with the double asterisks valid.
  - Expressions like 2^3 will be rejected because "^" is not a valid AEE operator.
- Parentheses must be used when performing negative exponentiation.
  - Example: 2**(-3)

Modulus

- Modulus is performed by the "%" operator.
  - Example: 5%3

Multiplication and Division

- Multiplication is performed by the "*" operator.
  - Example: 3*5

- Division is performed by the "/" operator.
  - Example: 2/3

More complex expressions can also be evaluated by the AEE. One example is (((2 ** (1 + 1)) + ((3 - 1) ** 2)) / ((4 / 2) % 3)). This expression contains all of the AEE's operations and is considered valid.

7. Glossary of terms

AEE- Arithmetic Expression Evaluator in C++; the name of the project outlined in the SAD document

EECS- the department of Electrical Engineering and Computer Science at the University of Kansas

EECS 348- class number for Software Engineering I at the University of Kansas

int- integer

KU- University of Kansas

SAD- Software Architecture Design document

UPEDU- Unified Process for Education; software development process that streamlines development

str- string

cin- standard input stream

Unix- a family of command-based operating systems which differ from macOS and Windows

Linux- a Unix operating system

8. FAQ

Q: What are the respective symbols for operations?

A: Respective symbols for operations are as follows: + add, - subtract, * multiplication, / division, ** exponentiation, % modulus.

Q: What is the largest number that I can input into the Arithmetic Expression Evaluator?

A: The largest number the AEE can handle is 1.8×10^{308} since this is the largest number a double type can hold in C++, and anything larger would overload the register size in the hardware.

Q: What is the best way to run this software?

A: The best way to run this software is to download the .exe file or put all the files from the Github repository (linked above) into the same folder and run it that way. This software is best to run on a Linux command line interface but works otherwise as well.

Q: Why was C++ the chosen language for this project?

A: C++ was used for the AEE because of closer connection to memory access compared to other languages that the team had experience with, and because this was the language that the course was focusing on.

Q: Does this program support redundant parentheses usage?

A: The program will handle redundant parentheses as long as all open parentheses have a closing parenthesis without changing the logic of the expression.