HEAAP Software

Arithmetic Expression Evaluator in C++

User's Manual

Version 1.1

Revision History

Date	Version	Description	Author
12/11/2024	1.0	Sections assigned and layout discussed.	All
12/12/2024	1.1	All sections finalized, changes made according to changes and bug fixes in implementation.	All

Table of Contents

Test Case

1. Purpose

The user manual for the arithmetic expression evaluator serves as an easy-to-understand guide on how to use the software. This manual guides the user on how to use the software effectively. It will explain how to install the software and input the expression by detailing the input syntax. Advanced features, troubleshooting, and examples of use will also be covered. A glossary and frequently asked questions section can be found at the end of the manual for convenience.

2. Introduction

This software functions as an arithmetic expression evaluator within the compiler of the new language, *L.* It takes a string from the user in the terminal, parses it, evaluates it, and returns the output, whether the output is the result of the arithmetic expression or an error. Possible operations include addition, subtraction, multiplication, division, modulus, and exponentiation.

To install this software,

1. Click on the link to the attached GitHub repository,
2. Click on the green Code button
3. Click Download Zip
4. Go to your Downloads folder
5. Right click on eecs348-project folder, click 7-zip, and click Extract here
6. Move the eecs348-project folder to the desktop

For Windows:

7. Open the "Command Prompt" application (can be easily accessed by searching in Windows search bar)
8. Navigate to the Desktop folder (type cd Desktop into the terminal)
9. Navigate to the eecs348-project folder (type cd eecs348-project into the terminal)
10. Type the command "make calculator" into the terminal and press the Enter key.
11. Type the command "./calculator.exe" into the terminal and press the Enter key.

For Unix-based

7. Open the "Terminal" application
8. Navigate to the Desktop (type cd Desktop into the terminal)
9. Navigate to the eecs348-project Folder
10. Type the command "make calculator" into the terminal and press the Enter Key.
11. Type the command "./calculator.exe" into the terminal and press the Enter key.

3. Getting started

Once the file is open and running, the command line will prompt the user to provide an expression, to view test cases, or to quit the program.

1. Entering an expression: Type entire expression (spaces allowed) into the field in the command line. Parentheses are supported by the calculator, as well as integers and six operators.
   Operators that can be used in the expression are:
   a. ** (exponentiation)
   b. * (multiplication)
   c. / (division)
   d. % (modulo)
   e. + (addition)
   f. – (subtraction)

After entering the full expression into the typing field, press the Enter key. If a valid expression is entered, the result will be output to the terminal. Read the integer in this bottom line to determine the result of the entered expression. In the case that the input was not a valid expression, an error message will appear describing why the expression was invalid, and the user will be prompted to input another expression. For more information on entering expressions, view the "Examples" section of the User Manual.

2. Viewing test cases: Type "t" into the field in the command line to view a list of test cases that then run on the calculator and show their pass/fail status.

3. Quitting the program: Type "q" into the field in the command line to quit the program.

4. Advanced features

The current version of the software includes all basic features which are described within the other sections of this document. Advanced features such as custom variables may be implemented and released in a future version of the software, in which case the user manual will be updated to accommodate these additions.

An additional feature added to this calculator is implied multiplication. The program parses numbers next to parentheses as implied multiplication; that is, the term inside the parentheses will be multiplied by the number adjacent to the parentheses. No sign is needed between the number and parentheses.

5. Troubleshooting

All errors are caught and handled by the program and are then relayed to the user through error messages. In the case of an error, the user should read the error message displayed in the command line and solve the error by entering a new, corrected expression when prompted. The most common errors are invalid expressions, in which the error message describes why the input is not a valid expression. Another error occurs when valid expressions are copy and pasted into the command line. This action may add invalid characters that the user does not see and will result in an error message. It is recommended that expressions are typed into the field instead.

6. Examples

The following are examples of expressions and images of what the input and output looks like in the command line.

Example 1: 3 + 6 – 10

```
Provide an expression to simplify, enter 't' to run test cases, or 'q' to quit:
3 + 6 - 10
-1
```

The result, which is found in the last line, is -1.

Example 2: 18 / (2 + 4)

```
Provide an expression to simplify, enter 't' to run test cases, or 'q' to quit:
18 / (2 + 4)
3
```

The result is 3.

Example 3: 4 * (3 + 2) % 7 – 1

```
Provide an expression to simplify, enter 't' to run test cases, or 'q' to quit:
4 * (3 + 2) % 7 - 1
5
```

The result is 5.

Example 4: 5 (4)

```
Provide an expression to simplify, enter 't' to run test cases, or 'q' to quit:
5(4)
20
```

The result is 20.

Example 5: 2 * (4 + 3 – 1

```
Provide an expression to simplify, enter 't' to run test cases, or 'q' to quit:
2 * (4 + 3 - 1
Error: Unmatched opening parenthesis.

Provide an expression to simplify, enter 't' to run test cases, or 'q' to quit:
2 * (4 + 3 - 1)
12
```

The result is an error with the message "Unmatched opening parenthesis." The expression is then re-entered with the corrections made, and the result is 12 for the valid expression.

7. Glossary of terms

Command Line - A text-based tool where you type commands to tell your computer what to do, like running programs or managing files

Compiler - A program that converts the code you write (in languages like C++ or Java) into a format the computer can understand and run

GitHub Repository - An online folder on GitHub where all the files and history of a project are stored, allowing people to work together on software and keep track of changes

Implied Multiplication - In programming or math, it's when you assume multiplication without writing the * sign, like when you write 2x instead of 2 * x

Make - A tool used to automate building a program. It looks at which parts of your project have changed and compiles only those, saving time during development

Parsing - The process of analyzing a string of symbols (such as code or data) according to the rules of a formal grammar. In software, parsing is used to interpret input, such as a source code file, and convert it into a format suitable for further processing.

Pass/Fail Status - A binary evaluation mechanism used in software engineering, often in testing, to indicate whether a particular process, task, or test case met the success criteria (pass) or failed to meet them (fail)

Terminal - A tool where you type text commands to directly control your computer, like opening files, running programs, or checking logs

8. FAQ

Q: Can I reuse the result from a previous calculation?

A: No. The calculator does not save previous results for reuse so you would need to manually include previous results in all new expressions.

Q: Can I enter decimals and/or fractions in my input expression?

A: As of now our calculator only supports integers to be entered as inputs. The usage of decimals and fractions may become available in updated versions of our calculator down the line.

Q: Are spaces allowed in my expression?

A: Yes. Spaces are allowed within our calculator and will not alter the expected output of the expression entered.

Q: My expression with a division operator seems to be giving the wrong result. Is this correct?

A: Because the calculator uses integer division, the results of division that occur in intermediate or final steps in the evaluation of the expression will be integers. C++ integer division will floor to the greatest integer below the number for positive values, and ceiling to the smallest integer above the number for negative values. The result of the expression is correct, although it may differ from the user's expected behavior of the program if they are unaware of the integer division.