17 Co Calculator

User's Manual

Version 1.0

Revision History

Date	Version	Description	Author
03/Dec/23	1.0	Initial version	17 Co. Team

Table of Contents

User's Manual

1. Purpose

The purpose of this document is to provide an easy-to-understand guide on how the software operates, how a user will interact with the program, a brief description of some of the advanced features our software provides, and how the 17 Co. Calculator handles any form of troubleshooting. To further assist users, the document includes examples, a glossary of terms, and an additional FAQ section at the end, offering comprehensive support and guidance for a smoother user experience. This ensures that users can easily find answers to common questions and better understand the terminology and functionalities of the software.

2. Introduction

The 17 Co. Calculator runs on a recursive solving algorithm that verifies each part of the expression through rigorous testing, then breaks the expression down into multiple simple expressions for solving. The program takes in a user's expression, and tests all the different ways the user's expression could be deemed invalid. If a valid expression is entered, the program will then make the expression into two vectors and adjust for any multi-digit numbers. The final part of the program uses a recursive function to calculate each part of the expression, using PEMDAS order of operations to find the next step to be solved. The output is a correct answer to the user's expression. To install the program, navigate to the project's GitHub and download the executable named **17CoCalculator**, then run the executable.

3. Getting started

Using the 17 Co. Calculator is fairly simple. Once the executable is running (instructions are in *1. Introduction*), you will then be prompted to provide input. This is where you will type the expression that you would like evaluated, or the word "exit" to terminate the program. Once you have entered either the expression or the keyword "exit", the program will either a) display the solved value, b) display an error, or c) terminate if "exit" is entered.

For a), the solved value will be displayed on the line under the line where you entered your expression. For b), an error will be displayed at the same place the solved value would have been displayed in the following format "Error – [error message here]" where [error message here] will display what error occurred. For example, you may receive a message such as "Error – Division by zero". This may be solved by providing an expression that does not divide by zero. More information can be found in *5. Troubleshooting*.

The program will only accept a single expression. It provides access to the use of the + (addition), - (subtraction), * (multiplication), / (division), % (modulo), and ^ (exponentiation) operators. It can also handle unary + and unary -. In order to use these operators, you must provide a value on both the left and right of the operator (with the exception of the unary operators). These values however may be nested parentheses with expressions within them. For example, (3^3)+3)/2 is a valid expression. The unary operators must just have a value on the right. For the purposes of this guide, an expression surrounded by parenthesis (e.g. (3+3)) is a value since it evaluates a single numeric constant. Ultimately, the entire expression should evaluate to a single numeric constant.

4. Advanced features

The main advanced feature of this calculator is the ability to use unary operators such as + and -. The simplest examples are that -(3) evaluates to -3 and +3 evaluates to 3. A more advanced example is that -(3)^(-(-+3)) evaluates to -27 as expected. What is happening here is that the two unary –'s in the power section evaluate to positive 3, and the unary +, as prescribed, has no major effect. Then we evaluate 3^3 which results in 27. Then the unary – causes the expression to evaluate to -27.

5. Troubleshooting

There are some common problems that might arise due to the behavior of the 17 Co. Calculator. The problems will be listed below with an example of an input that results in the problem, and then a corrected error of the input which fixes the problem. That will be followed by an explanation of what causes the problem and how it can be avoided.

"Invalid usage of operator + or -": +5+2- | +5+2-0

This problem arises when there is a missing expression on the right side of the addition or subtraction sign. In the example above, the subtraction sign does not have a subsequent number or expression. This is invalid since the calculator is trying to subtract a number that does not exist in the input. To prevent this problem, make sure there is always a number following addition or subtraction signs. This does not apply to addition or subtraction signs before numbers since those will be handled as unary operators.

*"Invalid usage of operator *, /, %, or ^"*: */5*2^ | 30/5*2^4*

This problem occurs if there is a missing expression on either side of a multiplication, division, modulo, or exponentiation operator. In the example above, both situations occur. There is no number preceding the division operator (/) and no expression following the exponentiation operator. The example above is fixed by adding numbers before and after the division and exponentiation operators respectively. To prevent this issue, ensure there are either numbers or parenthetical expressions before and after any multiplication, division, modulo, or exponentiation operators.

"Error - Division by zero": 17-2/(9%3) | 17-2/(9%3+1)

This error occurs when the calculator tries to divide an expression by the number 0. In the example above 9%3 evaluates to 0. So, then 2/0 is the next step evaluated resulting in the division by zero error. The fixed version of this adds one to the expression in the denominator so it evaluates to 1 instead of 0. In the fixed case, the program performs 15/1 and results in 1.

"Error - Invalid character(s) x": 15x7 | 15*7

The invalid character error occurs when an invalid character is entered as part of an expression. The error will say invalid characters, and then show the first invalid character in the expression. In the example above, an "x" is used as a multiplication operator despite it being an invalid character. This is fixed by changing the "x" character to the proper multiplication operator "*". To prevent this error from occurring, make sure used expressions only contain valid characters. Valid characters include: numeric constants, +, -, /, %, ^, (, and).

"Error - Root of a negative number": (-5)^(1/2) | -5^(1/2) *or* -(5)^(1/2)

The root of a negative number error occurs when the 17 Co. Calculator attempts to take the root of a negative number. This occurs because roots of negative numbers do not exist. This can be prevented by not taking the root of a negative number. If the intention is to take the negated version of the root of a number, you can put the expression that the root is being applied to in parenthesis. Or if it is the root of a singular numeric constant, the parenthesis can be removed all together, as the unary operator will be applied after the root.

"Error - Input must be provided": [Enter key is pressed with no input] | [Some input is provided]

This error occurs when the 17 Co. Calculator attempts to evaluate an expression which contains no characters. This happens when the enter key is pressed without anything being entered as an expression.

"Error - Input cannot be all spaces": [Spaces are the only characters in the expression] | [Any valid character is in the expression]

This error occurs when the 17 Co. Calculator attempts to evaluate an expression which only contains spaces. This happens when the input only has spaces and no valid characters. To prevent this, at least one valid character must be in the expression since the 17 Co. Calculator handles extra space characters.

6. Examples

The 17 Co. Calculator can evaluate multiple types of arithmetic expressions:

Addition (+)
Entered Expression: 3 + 4
17 Co. Calculator Output: 7

Subtraction (-)
Entered Expression: 10 – 5
17 Co. Calculator Output: 5

Multiplication (*)
Entered Expression: 6 * 7
17 Co. Calculator Output: 42

Division (/)
Entered Expression: 20 / 4
17 Co. Calculator Output: 5

Modulus (%)
Entered Expression: 10 % 3
17 Co. Calculator Output: 1

Exponential (^)
Entered Expression: 2^3
17 Co. Calculator Output: 8

Unary Plus (+)
Entered Expression: +5
17 Co. Calculator Output: 5

Unary Minus (-)
Entered Expression: -4
17 Co. Calculator Output: -4

Parentheses '()'
Entered Expression: 2 * (3+4)
17 Co. Calculator Output: 14

7. Glossary of terms

Additional information may be found in document D00002 *Project Glossary*.

8. FAQ

8.1.1 My expression generated an error. What should I do now?

After receiving an error, 17 Co. suggests that you revise your expression to resolve whatever error may have been generated. For example, if you have received a division by zero error, you should edit your expression to avoid dividing by zero. Sometimes that may be difficult to spot, but with proper mathematical inspection you should be able to resolve the error.

8.1.2 How do I run the program?

Running the program is a simple endeavor. All that is needed is for the executable file to run in the terminal. In Linux, this can be accomplished using the "./" command.

8.1.3 I have run the program. What do I do now?

Now that you have run the program, you may type in your mathematical expression to have it solved, or type "exit" to terminate the program. Once you have entered an expression, you will either see a result displayed or an error. Either way, you will then be prompted for input again in order to easily allow you to enter a new expression.

8.1.4 What are the valid operators?

Valid operators are + (addition), - (subtraction), * (multiplication), / (division), % (modulo), and ^ (exponentiation). It can also handle unary + and unary -.

8.1.5 Can the program handle expressions with parentheses?

Yes. The program can handle expressions with parentheses, including nested parentheses.