QuickMath.io

User's Manual

Version 1.0

Revision History

Date	Version	Description	Author
01/12/2023	0.1	Added information	David, Vinny
02/12/2023	1.0	Reviewed and finalized material	David, Vinny, Omar, Owen, Jamie, Tatum

Table of Contents

User Manual

1. Purpose

This document serves to show prospective users how to use our application, answer any questions, and troubleshoot any issues. This document is loosely broken up into the following pieces: acquiring the application, using the application, troubleshooting any issues, a glossary for commonly used abbreviations, and frequently asked questions.

2. Introduction

The product QuickMath.io is a command-line interface to perform arithmetic operations. The product parses the input given in the command-line and evaluates the expression using the Shunting-Yard algorithm, respecting the precedence of operators, and parenthesis.

The features of the product are:

1. Operators support: The operators supported are:
    a. + (Addition)
    b. - (Subtraction)
    c. * (Multiplication)
    d. / (Division)
    e. % (Modulo)
    f. ^ (Exponential)
2. Unary operator support: "+" and "-" are also supported as unary operators.
3. Parenthesis handling: Multiple nested layers of parenthesis are supported. If there is a mismatch between the parenthesis, for example, no complete pair, then an error is reported to the user.
4. Numeric constants: "pi" and "e" are supported as numeric constants
5. Error handling: The program is robust to faulty errors and handles and reports various errors such as incomplete parenthesis, extraneous operators, extraneous numbers.
6. Float handling: The program also supports floating point numbers as the inputs.

To install the product, do the following steps

1. Run **`git pull https://github.com/realVinayak/EECS348_Project`**
2. Go into the code directory by running **`cd ./code/`**
3. Run **`make`**
4. Run the compiled executable by running **`./main.o`**

3. Getting started

Once the steps 1 through 4 are completed from the installation instructions, the command-line interface will start asking for an input as "Enter an arithmetic expression: ", as shown below.

```
Enter an arithmetic expression:
```

3.1 Guide to use operators

3.1.2 Binary Operators

Binary operators are operators which need two operands to compute a result. The following binary operators are supported:

1. + (Addition)
2. - (Subtraction)
3. * (Multiplication)
4. / (Division)
5. % (Modulo)

6. ^ (Exponential)

To use a binary operator, simply enter two operands, and the binary operator between the operands. The operands can be numeric values, numeric constants, or expressions wrapped in parentheses. For example, the following uses of binary operands are valid. Additionally, ** gets treated as ^ (Exponentiation).

1. 90 + 20
2. 90 + 30 – 20
3. (90 + 30) / (20 – 9)
4. 20 * pi

3.1.2 Unary Operators

Unary operators are those operators which act on just on operand. The unary operators supported are

1. + (positive)
2. - (negation)

To use a unary operator, simply enter an operand succeeded by a unary operator. Similar to binary operators, the operand can be a numeric value, numeric constant or expression wrapped in parentheses. For example, the following uses of unary operands are valid

1. +90
2. -90
3. +(-20)
4. -(-30 - (-20))

3.1.3 Arithmetic Errors

The following arithmetic errors are recognized and caught

1. Division by zero: If the denominator is zero (or computes to zero), "Division by zero" error is raised and shown to the user
2. Modulus by zero: If the second operand is zero (or computes to zero), "Modulus by zero" error is raised and shown to the user

3.1.4 Precedence of Operators

The following precedence of operators is followed (in decreasing order of precedence), given there is no parentheses:

1. Unary positive (+), Unary negation (-)
2. Exponentiation (^)
3. Multiplication (*), Division (/), Modulus (%)
4. Addition (+), Subtraction (-)

3.2 Guide to use parentheses

Parentheses are exhaustively supported. If there is an error in parsing the expression due to mismatch in opening and closing pairs, the error "Error: Mismatched parentheses in the expression." is reported.

To use parentheses, simply enclose any valid arithmetic expression inside of a parentheses pair. The expression can in turn contain parentheses, and hence nested parentheses are supported. The following usages of parentheses are supported:

1. (+20)
2. +(20 – 90)
3. – (– 20)
4. +(20 - (+90 – 20 + (20 – 2))

Examples of invalid usage of parentheses are:

1. (20
2. -90)
3. ()

3.3 Guide to use numeric constants

Well-known numeric constants are supported. The currently supported constants are

1. π - Treated as 3.14159. To use this, type "pi"
2. e – Tread as 2.71828. To use this, type "e"

3.4 Guide to interpreting the results

If the expression is valid, the output is "Result:" followed by the result of evaluating the expression.
If the expression is invalid, the following errors are shown

1. Error: Mismatched parentheses in the expression.
    a. Refer Section 3.2
2. Error: Division by zero
    a. Refer Section 3.1.3
3. Error: Modulus by zero
    a. Refer Section 3.1.3
4. Error: Input is invalid
    a. Raised if the input is invalid such as if it contains invalid characters or if the . For example,
        i. 20 &30 + &20
        ii. 20 & 10
        iii. 2 0
        iv. 20 +
        v. 90 (20 – 30)

3.5 Guide to entering expressions

The input is whitespace agnostic. Hence, any whitespace can be entered in between recognized tokens (numbers, operators, numeric constants) and the expression will still be evaluated. For example, the following expressions of "pi + 20 – e * (20 + (+20))" are equivalent

1. pi + 20 – e * (20 + (+ 20))
2. pi+20– e*(20 + (+20))

However, the following expressions are not equivalent

1. p i + 20 – e * (2 0 + (+ 20))
2. pi + 20 – e * (2 0 + (+ 20))

4. Advanced features

Our product contains no advanced features. This application will calculate arithmetic operations.

5. Troubleshooting

This section is organized as follows: The issue you may be having is addressed by text that follows it.

You have more parentheses than necessary: Remove the extraneous symbols.
You do not have enough parentheses: Add symbols until you have the same number of open and close parentheses.
You have too many operators: Remove the excess operators.
You got an invalid result due to division: You may have attempted to divide by zero. If this is the case, remove the quotient.
You got a weird answer in general: This could have been caused by many things, as listed: You may have included an extraneous operand in your calculations, which should be removed.
The input was invalid for any other reason: You attempted to use a symbol that is unrecognized. Remove it.

6. Examples

6.1 Examples of Valid Expressions

1. Addition: 6 + 10

```
Enter an arithmetic expression: 6 + 10
Result: 16
```

2. Subtraction with Parentheses: 10 - (9 – 3)

```
Enter an arithmetic expression: 10 - (9 - 3)
Result: 4
```

3. Multiplication and Division: 90 * 10 / 3

```
Enter an arithmetic expression: 90 * 10 / 3
Result: 300
```

4. Exponentiation: 20 ^ 3

```
Enter an arithmetic expression: 20 ^ 3
Result: 8000
```

5. Mixed Operators: 10 * 8 % 3 – 1

```
Enter an arithmetic expression:  10 * 8 % 3 - 1
Result: 1
```

6. Complex Addition with Extraneous Parentheses: (((10 + 29))) + (((11 + 22)) + ((90 + 20)))

```
Enter an arithmetic expression:  (((10 + 29))) + (((11 + 22)) + ((90 + 20)))
Result: 182
```

7. Mixed Operators with Extraneous Parentheses: ((5 * 2) - ((3 / 1) + ((4 % 3))))

```
Enter an arithmetic expression: ((5 * 2) - ((3 / 1) + ((4 % 3))))
Result: 6
```

8. Nested Parentheses with Exponents: (((2 ^ (1 + 1)) + ((3 - 1) ^ 2)) / ((4 / 2) % 3))

```
Enter an arithmetic expression: (((2 ^ (1 + 1)) + ((3 - 1) ^ 2)) / ((4 / 2) % 3))
Result: 4
```

9. Combination of Extraneous and Necessary Parentheses: (((((5 - 3))) * (((2 + 1))) + ((2 * 3))))

```
Enter an arithmetic expression: (((((5 - 3))) * (((2 + 1))) + ((2 * 3))))
Result: 12
```

10. Extraneous Parentheses with Division: ((9 + 6)) / ((3 * 1) / (((2 + 2))) - 1)

```
Enter an arithmetic expression:  ((9 + 6)) / ((3 * 1) / (((2 + 2))) - 1)
Result: -60
```

11. Combining Unary Operators with Arithmetic Operations: +(-2) * (-3) - ((-4) / (+5))

```
Enter an arithmetic expression: +(-2) * (-3) - ((-4) / (+5))
Result: 6.8
```

12. Unary Negation and Addition in Parentheses: -(+1) + (+2)

```
Enter an arithmetic expression: -(+1) + (+2)
Result: 1
```

13. Negation and Addition with Negated Parentheses: -(-(-3)) + (-4) + (+5)

```
Enter an arithmetic expression: -(-(-3)) + (-4) + (+5)
Result: -2
```

14. Unary Negation and Exponentiation: +2 ^ (-3)

```
Enter an arithmetic expression: +2 ^ (-3)
Result: 0.125
```

15. Combining Unary Operators with Parentheses: -(+2) * (+3) - (-4) / (-5)

```
Enter an arithmetic expression:  -(+2) * (+3) - (-4) / (-5)
Result: -6.8
```

6.2 Examples of Invalid Expressions

1. Unmatched Parentheses: 2 * (4 + 3 – 1

```
Enter an arithmetic expression: 2 * (4 + 3 - 1
Error: Mismatched parentheses in the expression.
```

2. Operators Without Operands: * 5 + 2

```
Enter an arithmetic expression: * 5 + 2
Result: Error: Input is invalid.
```

3. Incorrect Operator Usage: 4 / 0

```
Enter an arithmetic expression: 4 / 0
Result: Error: Division by zero
```

4. Missing Operator: 5 (2 + 3)

```
Enter an arithmetic expression: 5 (2 + 3)
Error: Input is invalid
```

5. Invalid Characters: 7 & 3

```
Enter an arithmetic expression: 7 & 3
Error: Input is invalid
```

6. Mismatched Parentheses: (((3 + 4) - 2) + (1)

```
Enter an arithmetic expression: (((3 + 4) - 2) + (1)
Error: Mismatched parentheses in the expression.
```

7. Invalid Operator Usage: ((5 + 2) / (3 * 0))

```
Enter an arithmetic expression: ((5 + 2) / (3 * 0))
Result: Error: Division by zero
```

8. Invalid Operator Sequence: ((2 -) 1 + 3)

```
Enter an arithmetic expression: ((2 -) 1 + 3)
Result: Error: Input is invalid.
```

9. Missing Operand: ((4 * 2) + (-))

```
Enter an arithmetic expression:  ((4 * 2) + ( - ))
Error: Input is invalid
```

10. Invalid Characters: ((7 * 3) @ 2)

```
Enter an arithmetic expression: ((7 * 3) @ 2)
Error: Input is invalid
```

7. Glossary of terms

FAQ – Frequently Asked Questions

8. FAQ

In this section, we address some frequently asked questions.

Q: Does this calculator come with a graphing function?

A: No.

Q: Will the calculator operate on variables?

A: No.

Q: Does this calculator perform differentiation?

A: No.

Q: Does this calculator perform integration?

A: No.

Q: If I download this calculator, will my information be funneled to a data center?

A: No, but even if it did, your data is likely already in a datacenter.

If we receive a large influx of any other questions, we will add them to the FAQ section.