SimpleMathParser (2.0)

Kcits970

December 25, 2021

Sections

Section 1. Description

The purpose of the program ‘SimpleMathParser’ is to parse mathematical expressions including constants and variables. It provides a menu-driven interface to input expressions, edit variables, and view calculation history.

Special thanks to the StackOverFlow user Boann for providing a useful example of a basic math expression compiler[1].

The program is written in C language, and compiled using Microsoft Visual C++ 14.2. Attempts to compile the given source code with a different compiler may result in unexpected or undefined behavior.

[1] Boann's answer to "How to evaluate a math expression given in string form?"

Section 2. Features

The features of ‘SimpleMathParser’ can be divided into 3 main parts, which are expression input, variables, and calculation history.

Expressions

The main options are prompted at program launch.

```
>>MATH EXPRESSION PARSER: VERSION 2

    <MAIN OPTIONS>
INPUT MATH EXPRESSION ----------- A
EDIT EXPRESSION VARIABLES ------- B
VIEW EXPRESSION DETAILS --------- C
DELETE EXPRESSION --------------- D
VIEW HISTORY -------------------- E
CLEAR HISTORY ------------------- F
EXIT ---------------------------- Q
>>
```

To calculate an expression, select the `INPUT MATH EXPRESSION` option. From there, the user can input an math expression in string form, and the program will start to parse the given input.

The program is able to parse the following terms.

Numeric Constants	1	2	12.34	56.78901	etc	
Operators	+	-	*	/	^	
Named Constants	e	pi(π)	phi(ϕ)			
Functions	sqrt	cbrt	sin	cos	tan	ln

It should be noted that the parser is case-sensitive. This means that `sqrt2` will be parsed to 1.414..., but `SQRT2` will fail to parse. Below are some examples of what gets correctly parsed and what doesn’t.

Valid Input

```
    <EXPRESSION INPUT PROMPT>
ENTER A MATH EXPRESSION: cos(sin pi)
'cos(sin pi)' EVALUATES TO '1.000000'
>>
```

Invalid Input: Trailing Text

```
    <EXPRESSION INPUT PROMPT>
ENTER A MATH EXPRESSION: cos pi with some text
UNABLE TO PARSE FULL EXPRESSION: UNREACHABLE 'with some text' AT INDEX 7
'cos pi with some text' EVALUATES TO '-1.000000'
```

```
>>
```

Invalid Input: Capitalized Function

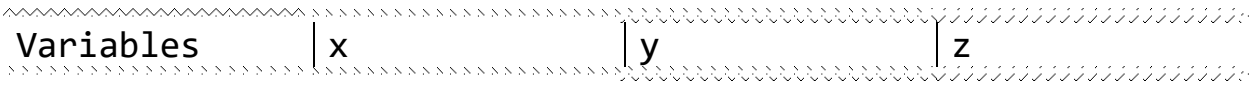
```
    <EXPRESSION INPUT PROMPT>
ENTER A MATH EXPRESSION: COS PI - SQRT 5
UNEXPECTED CHARACTER 'C' AT INDEX 0
UNABLE TO PARSE FULL EXPRESSION: UNREACHABLE 'COS PI - SQRT 5' AT INDEX 0
'COS PI - SQRT 5' EVALUATES TO '0.000000'
>>
```

Invalid Input: Capitalized Constants

```
    <EXPRESSION INPUT PROMPT>
ENTER A MATH EXPRESSION: E*PI*PHI
UNEXPECTED CHARACTER 'E' AT INDEX 0
UNABLE TO PARSE FULL EXPRESSION: UNREACHABLE 'E*PI*PHI' AT INDEX 0
'E*PI*PHI' EVALUATES TO '0.000000'
>>
```

Variables

Aside from expressions with constant values, the program also allows variables. Variables are marked with “x”, “y”, or “z”. In another words, up to 3 different variables can be used.



When the parser encounters a variable during the parse, it will prompt the user to enter a value for the specified variable. If multiple variables are used, the parser will ask for the values in sequential order.

```
    <EXPRESSION INPUT PROMPT>
ENTER A MATH EXPRESSION: ln(x * z * y)
ENTER A VALUE FOR VARIABLE 'x': e
ENTER A VALUE FOR VARIABLE 'z': 1
ENTER A VALUE FOR VARIABLE 'y': e^5
'ln(x * z * y)' EVALUATES TO '6.000000'
>>
```

Variables are very similar to expressions, but they do not allow variables inside them. This means that variables cannot contain variables. If the parser encounters a variable while parsing a variable, the parser will ignore the variable character and raise an error message.

```
    <EXPRESSION INPUT PROMPT>
ENTER A MATH EXPRESSION: cbrt(x + 10)
ENTER A VALUE FOR VARIABLE 'x': 2 - y
REACHED END OF STRING WHILE PARSING
UNDEFINED CONSTANT/VARIABLE/FUNCTION NAME 'y' AT INDEX 4
```

```
'cbrt(x + 10)' EVALUATES TO '2.289428'
>>
```

Variables of an expression can be changed by selecting the `EDIT EXPRESSION VARIABLES` option.

```
    <MAIN OPTIONS>
INPUT MATH EXPRESSION ----------- A
EDIT EXPRESSION VARIABLES ------- B
VIEW EXPRESSION DETAILS --------- C
DELETE EXPRESSION --------------- D
VIEW HISTORY -------------------- E
CLEAR HISTORY ------------------- F
EXIT ---------------------------- Q
>>B

    <EXPRESSION SELECTION PROMPT>
EXPRESSION 001: 'cbrt(x + 10)'
EXPRESSION 002: 'cbrt x'
EXPRESSION 003: 'ln(x * z * y)'
EXPRESSION 004: 'ln x'
EXPRESSION 005: 'E*PI*PHI'
EXPRESSION 006: 'COS PI - SQRT 5'
EXPRESSION 007: 'cos pi with some text'
EXPRESSION 008: 'cos(sin pi)'
SELECT AN EXPRESSION: 3

    <VARIABLE EDIT PROMPT>
SELECTED EXPRESSION: 'ln(x * z * y)'
VARIABLE X: 2 * cos pi
VARIABLE Y: 100
VARIABLE Z: -e
'ln(x * z * y)' EVALUATES TO '6.298317'
>>
```

After the user chooses an expression from the provided list, the parser will ask for new inputs of each variable and recalculate the result.

Calculation History

The calculations are kept in a linked-list storage. This storage is called a `history list`[2]. The maximum size of this list is set to 100. Below are examples of some actions that either use or modify this list.

VIEW EXPRESSION DETAILS(C)

```
    <EXPRESSION SELECTION PROMPT>
EXPRESSION 001: 'cbrt(x + 10)'
EXPRESSION 002: 'ln(x * z * y)'
```

[2] In the source code, it is referred as the “expressionList”.

```
EXPRESSION 003: 'ln x'
EXPRESSION 004: 'E*PI*PHI'
EXPRESSION 005: 'COS PI - SQRT 5'
EXPRESSION 006: 'cos pi with some text'
EXPRESSION 007: 'cos(sin pi)'
SELECT AN EXPRESSION : 2

    <EXPRESSION DETAILS>
FULL EXPRESSION: 'ln(x * z * y)'
VARIABLE X: '2 * cos pi'
VARIABLE Y: '100'
VARIABLE Z: '-e'
RESULT: 6.298317
>>
```

DELETE EXPRESSION(D)

```
    <EXPRESSION SELECTION PROMPT>
EXPRESSION 001: 'cbrt(x + 10)'
EXPRESSION 002: 'ln(x * z * y)'
EXPRESSION 003: 'ln x'
EXPRESSION 004: 'E*PI*PHI'
EXPRESSION 005: 'COS PI - SQRT 5'
EXPRESSION 006: 'cos pi with some text'
EXPRESSION 007: 'cos(sin pi)'
SELECT AN EXPRESSION : 6
DELETED
>>
```

CLEAR HISTORY(F)

```
    <MAIN OPTIONS>
INPUT MATH EXPRESSION ----------- A
EDIT EXPRESSION VARIABLES ------- B
VIEW EXPRESSION DETAILS --------- C
DELETE EXPRESSION --------------- D
VIEW HISTORY -------------------- E
CLEAR HISTORY ------------------- F
EXIT ---------------------------- Q
>>F
SUCCESSFULLY CLEARED HISTORY
>>
```

Section 3. Program Execution

Execution in Windows Terminal: