Group 32

Arithmetic Expression Evaluator in C++

Test Case

Version <1.1>

Derek Norton, Ryan Grimsley, Jordan Burns, Manu Redd, Priyatam Nuney

Revision History

Date	Version	Description	Author
<01/12/23>	<1.0>	Edited purpose	Ryan G
<03/12/23>	<1.1>	Added and edited test case table	Derek N, Ryan G, Jordan B.

Table of Contents

1. Purpose 4

2. Test cases 4-6

3. Program Screenshots 6-7

Test Case

1. Purpose

This Test Case Specification document for the Arithmetic Expression Evaluator in C++ defined test cases for the Arithmetic Expression Evaluator that should be tested. These test cases are designed to find faults in the code by testing the boundary values as well as nominal values.

2. Test cases

Test Case ID	Test Case Description	Test Data	Expected Results	Actual Results	Pass/Fail
Add-101	Test addition operator	2 + 2	4	4	Pass
Add-102	Test addition operator with parentheses (split)	(2) + (2)	4	4	Pass
Add-103	Test addition with parentheses	(2+2)	4	4	Pass
Add-104	Test addition with multiple parentheses	(2+((2)))	4	4	Pass
Add-105	Test addition with extra unary negation	1 + --1	2	2	Pass
Add-106	Test addition with extra unary negation with parenthesis	1 + -(-1)	2	2	Pass
Add-107	Test addition with extra unary addition	1 + ++1	2	2	Pass
Sub-101	Test subtraction operator	10 - 5	5	5	Pass
Sub-102	Test subtraction operator with parentheses (split)	(10) - (5)	5	5	Pass
Sun-103	Test subtraction operator with parentheses	(10 - 5)	5	5	Pass
Sub-104	Test subtraction operator with multiple parentheses	(((10)) - (5))	5	5	Pass
Sub-105	Test negation operator	-(13 + 2)	-15	-15	Pass
Sub-106	Test negative values being added to positive values (no parentheses)	-1 + 1	0	0	Pass
Sub-107	Test negative values being added to positive values (inside parentheses)	(-1) + 1	0	0	Pass
Sub-108	Test negative values being added to positive values (outside parentheses)	-(1) + 1	0	0	Pass
Sub-109	Test negative values subtracted from negative values (with parentheses)	(-(12) - (6))	-18	-18	Pass
Sub-110	Test negative values subtracted from negative	-12 - 6	-18	-18	Pass

	values (no parentheses)				
Sub-111	Test subtraction operator with extra unary negation	1 - --1	0	0	Pass

Mult-101	Test multiplication operator	15 * 2	30	30	Pass
Mult-102	Test multiplication operator (inside parentheses)	(15*2)	30	30	Pass
Mult-103	Test multiplication operator (outside parentheses)	(15) * 2	30	30	Pass
Mult-104	Test negative value times negative value = positive value	-4 * -4	16	16	Pass
Mult-105	Test negative value times negative value = positive value (with parentheses)	-(4) * (-4)	16	16	Pass
Mult-106	Test multiplication times 0	15 * 0	0	0	Pass

Div-101	Test division operator	100 / 25	4	4	Pass
Div-102	Test division operator (inside parentheses)	(100 / 25)	4	4	Pass
Div-103	Test division operator (split parentheses)	(100) / 25	4	4	Pass
Div-104	Test divide by zero error	1 / 0	Error	Error	Pass
Div-105	Test divide by zero error (with parentheses)	(44) / 0	Error	Error	Pass
Div-106	Test divide by fraction (in parentheses)	(10) / (½)	20	20	Pass
Div-107	Test multiply by division	30 * (1/2)	20	20	Pass
Div-108	Test divide with unary negation	30 / -3	-10	-10	Pass

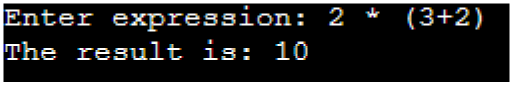
Mod-101	Test modulo operator	10%3	1	1	Pass
Mod-102	Test modulo fraction	100%(9/2)	1	1	Pass
Mod-103	Test modulo fraction (no parentheses)	100% 1/2	0	0	Pass
Mod-104	Test modulo operator with unary negation	100 % -3	-2	-2	Pass
Mod-105	Test modulo operator with unary addition	100 % +3	1	1	Pass

Exp-101	Test exponentiation operator	2 ^ 3	8	8	Pass
Exp-102	Test exponentiation operator with power in parenthesis	2 ^ (3)	8	8	Pass

Exp-103	Test exponentiation operator with base in parenthesis	(2) ^ 3	8	8	Pass
Exp-104	Test exponentiation operator with unary negation	2 ^ -3	.125	.125	Pass
Exp-105	Test exponentiation operator with unary negation in parenthesis	2 ^ (-3)	.125	.125	Pass
Exp-106	Test exponentiation operator with unary addition	2 ^ +3	8	8	Pass
Exp-107	Test exponentiation operator with unary addition with parenthesis	2 ^ (+3)	8	8	Pass

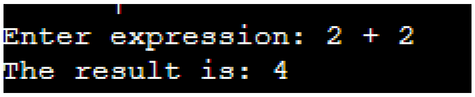
3. Example Screenshots

2 * (3+2) = 10



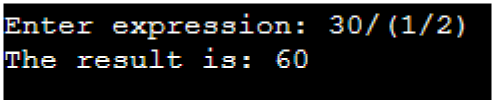
```
Enter expression: 2 * (3+2)
The result is: 10
```

2 + 2 = 4



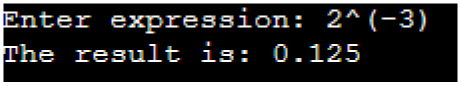
```
Enter expression: 2 + 2
The result is: 4
```

30/ (½) = 60



```
Enter expression: 30/(1/2)
The result is: 60
```

2^ (-3) = 0.125



```
Enter expression: 2^(-3)
The result is: 0.125
```