

Calculator

Naho Yokoyama

Overview

This program implements a calculator. When a user inputs a mathematical expression, the program will calculate and returns its value. This program supports calculation including addition, subtraction, multiplication, division, the use of parenthesis, absolute value function, integer value function, and round function.

How this program works



The diagram illustrates the state of a calculator's token stack and display after step 3. The token stack contains the tokens: '(', 'abs', '(', '3.5', '-', '5.5', ')', '+', '10', '*', '5', ')'. The display shows '-2.0'. A blue arrow points to the '+' token in the stack.

Diagram illustrating the evaluation of the postfix expression `3.5 - 5.5) + 10 * 5)` using a stack.

The expression is shown in a sequence of boxes: `(`, `abs`, `(`, `3.5`, `-`, `5.5`, `)`, `+`, `10`, `*`, `5`, `)`. A blue arrow points to the closing parenthesis `)` after `5.5`, indicating the current token being processed.

Steps:

1. Push all tokens to the stack until we have "*" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
 -3.1 multiplication & division
 -3.2 addition & subtraction
 -3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

Diagram illustrating the stack state during function evaluation:

The stack contains `-2.0` and `abs`. A bracket indicates the result of the function is `2.0`.

For calculating functions, we seek the token at the top of the stack to see if there is a function.

Pop it if there is

The diagram shows a sequence of tokens in a stack: (, abs, (, 3.5, -, 5.5,), +, 10, *, 5,). A blue arrow points to the closing parenthesis token ')'.

Diagram illustrating the state of a postfix calculator. The input tokens are: (abs (3.5 - 5.5) + 10 ^ 5). A blue arrow points to the '+' token, indicating it is the current operator being processed. Below the tokens, a stack is shown with '4.0' at the bottom and '2.0' above it. The stack is labeled 'Stack'.

Diagram illustrating the stack operations for the expression $((abs ((3.5 - 5.5)) + * 10) * 5)$:

The stack initially contains: $($, $($, abs , $($, 3.5 , $-$, 5.5 , $)$, $+$, 10 , $*$, 5 , $)$.

Step 1: Push all tokens to the stack until we have "(" token.

Step 2: Pop all tokens from the stack until we have "(" token.

Step 3: Calculate the value of the sequence of popped tokens:

- $\rightarrow 3.1$ multiplication & division
- $\rightarrow 3.2$ addition & subtraction
- $\rightarrow 3.3$ functions

Step 4: Create a token of the calculated value and push to the stack.

Step 5: Repeat steps 1-4 until we go through all of the tokens.

Diagram showing the stack after step 4:

10
+
2.0
(

Stack

The diagram shows a sequence of tokens in boxes: (, abs, (, 3.5, -, 5.5,), +, 10, *, 5,). A blue arrow points to the '*' token. Below the tokens, a stack is shown with the following elements from bottom to top: (, 2.0, +, 10, *. The word "Stack" is written below the stack.

((abs (3.5 - 5.5) + 10 * 5)

↑

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
 --3.1 multiplication & division
 --3.2 addition & subtraction
 --3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

5
*
10
+
2.0
(

Stack

Calculator display: (| abs | (| 3.5 | - | 5.5 |) | + | 10 | * | 5 |)

Steps:

- Push all tokens to the stack until we have ")" token
- Pop all tokens from the stack until we have "(" token
- Calculate the value of the sequence of popped tokens
 →3.1 multiplication & division
 →3.2 addition & subtraction
 →3.3 fractions
- Create a token of the calculated value and push to the stack
- Repeat steps 1-4 until we go through all of the tokens

Stack

Steps:

1. Push all tokens from the stock until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
 → 3.2 multiplication & division
 → 3.3 addition & subtraction
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

Stack

Diagram illustrating the evaluation of the postfix expression $3.5 \times 5.5 + 10 \times 5$ using a stack:

The input tokens are: (, abs, (, 3.5, -, 5.5,), +, 10, *, 5,).

The stack operations are shown in the table below:

Token	Stack
((
abs	(abs
((abs (
3.5	(abs (3.5
-	(abs (3.5 -
5.5	(abs (3.5 - 5.5
)	(abs (
+	(abs (+
10	(abs (+ 10
*	(abs (+ 10 *
5	(abs (+ 10 * 5
)	(abs

The final result is 52.0.

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
 → 3.1 multiplication & division
 → 3.2 addition & subtraction
 → 3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

Stack: 52.0

19

Steps:

1. Push all tokens to the stack until we have ")" token
2. Pop all tokens from the stack until we have "(" token
3. Calculate the value of the sequence of popped tokens
 → 3.1 multiplication & division
 → 3.2 addition & subtraction
 → 3.3 functions
4. Create a token of the calculated value and push to the stack
5. Repeat steps 1-4 until we go through all of the tokens

Stack: 52.0 Answer!

20

Input's tolerance

Allow

- negative numbers (ex. "-1+-5" will be evaluated as -6)
- use of more parenthesis than necessary (ex. "((((((5+5))))))")
- too many open parenthesis (outer parenthesis will be ignored, ex. "(3+5*2" becomes 13)
 → "eval" function to calculated expected answer will returns an error though

Not allow

- division by zero
- space between characters (ex. "1 + 5")
- incomplete input of decimals (ex. ".5")
- Too many close parenthesis (ex. "3+5)")