Assignment 4 – Calc Report Template

Nithin Duvvuru

CSE 13S – Winter 24

Purpose

Audience for this section: Pretend that you are working in industry, and write this paragraph for your boss. You are answering the basic question, "What does this thing do?". This section can be short. A single paragraph is okay.

Do not just copy the assignment PDF to complete this section, use your own words.

This assignment asks us to build a calculator in C using Reverse Polish Notation. It focuses on implementing mathematical functions like sine, cosine, and tangent, using ADTs like stacks and function pointers. We also use command-line options and error handling to make the code run smoothly and make it easy to debug.

Questions

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader's life easier, please do not remove the questions, and simply put your answers below the text of each question.

- Are there any cases where our sin or cosine formulae can't be used? How can we avoid this?

  There are two cases where the implemented sine or cosine formulas may not provide accurate results. It is when the input is very large or very small. To avoid this, we can make sure the user inputs a number within a reasonable range and create an error if not within that range.

- What ways (other than changing epsilon) can we use to make our results more accurate? [1]

  Using angle reduction techniques to limit input angles within specific ranges like between 0 and 2π for the trig functions can minimize errors.

- What does it mean to normalize input? What would happen if you didn't?

  Normalizing input makes sure the user inputs a number within a specific range. Without normalizing inputs, the program could output unintended results because of the way certain functions are calculated.

- How would you handle the expression 321+? What should the output be? How can we make this a more understandable RPN expression?

  The 2 and 1 should be added to 3 so the output should be a stack with values 3 and 3. To make this more understandable, I would add spaces in between all the numbers and operators.

[1]hint: Use trig identities

- Does RPN need parenthesis? Why or why not?

  RPN does not need parenthesis because the order of operations is consistent with the order of how the elements were typed in.

Testing

List what you will do to test your code. Make sure this is comprehensive. [2] Be sure to test inputs with delays and a wide range of files/characters.

First I would test each of the calculator functions with positive numbers, negative numbers, zero, and decimals. I would also test edge cases and see if the program handles all errors properly.

How to Use the Program

Audience: Write this section for the user of your program. You are answering the basic question, "How do I use this thing?". Don't copy the assignment exactly; explain this in your own words. This section will be longer for a more complicated program and shorter for a less complicated program. You should show how to compile and run your program. You should also describe any optional flags or inputs that your program uses, and what they do.

To show "code font" text within a paragraph, you can use `\lstinline{}`, which will look like this: `text`.

For a code block, use `\begin{lstlisting}` and `\end{lstlisting}`, which will look like this:

```
Here is some code in lstlisting.
```

And if you want a box around the code text, then use `\begin{lstlisting}[frame=single]` and `\end{ lstlisting}`
which will look like this:

```
Here is some framed code (lstlisting) text.
```

Want to make a footnote? Here's how.[3]

Do you need to cite a reference? You do that by putting the reference in the file `bibtex.bib`, and then you cite your reference like this[1][2][3].

While the calculator is running, you can enter expressions to evaluate. The calculator supports basic arithmetic operations such as addition, subtraction, multiplication, and division. It also supports mathematical functions like sine, cosine, tangent, absolute value, and square root. You can enter expressions in Reverse Polish Notation. The calculator evaluates the expressions and displays the result on the screen. You can continue entering expressions until you exit the program.

Program Design

Audience: Write this section for someone who will maintain your program. In industry you maintain your own programs, and so your audience could be future you! List the main data structures and the main algorithms. You are answering the basic question, "How is this thing organized so that I can have a chance of fixing it?". This section will be longer for a more complicated program and shorter for a less complicated program.

[2]This question is a whole lot more vague than it has been the last few assignments. Continue to answer it with the same level of detail and thought.

[3]This is my footnote.

The main data structure is the global stack which manages all the users inputs using the Last in, first out structure. The main algorithm we will use to handle user input is the reverse polish method which will be used to calculate the output of the users input.

Pseudocode

Give the reader a top down description of your code! How will you break it down? What features will your code have? How will you implement each function.

First, code will handle user input by either putting it in the stack to giving an error if it is an incorrect input.
Second, the code will parse through the user input and order it into the stack.
Lastly, the code will output the evaluated expression

Function Descriptions

For each function in your program, you will need to explain your thought process. This means doing the following

- The inputs of every function (even if it's not a parameter)
- The outputs of every function (even if it's not the return value)
- The purpose of each function, a brief description about a sentence long.
- For more complicated functions, include pseudocode that describes how the function works
- For more complicated functions, also include a description of your decision making process; why you chose to use any data structures or control flows that you did.

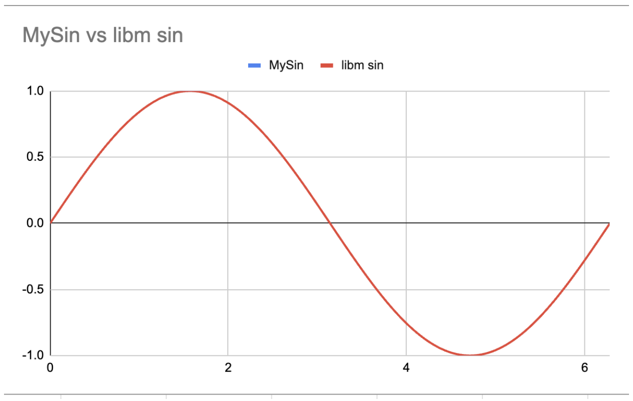
Do not simply use your code to describe this. This section should be readable to a person with little to no code knowledge. **DO NOT JUST PUT THE FUNCTION SIGNATURES HERE. MORE EXPLANATION IS REQUIRED.**

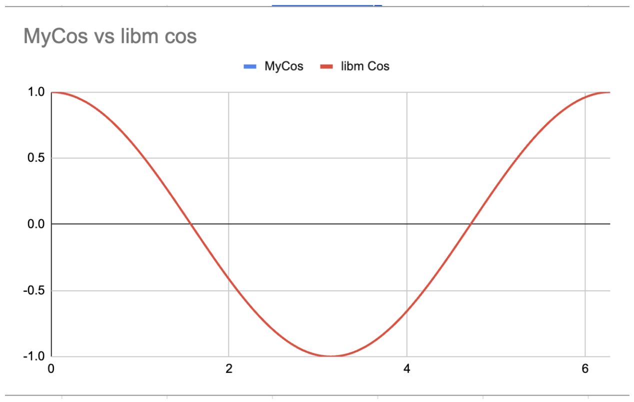
parse expression()
no inputs, or outputs
takes the string input from the user and adds all the elements to the stack and prints out errors when necessary

all the stack ADT functions
stack push, takes a double and puts it at the top of the stack. stack peek, returns copy of the element at the top of the stack. stack pop, removes the element at the top of the stack and returns it. stack clear, loops through the stack and removes all elements. stack print, loops through the stack and prints all elements in it.

abs, takes in a double and if it is less than 0, returns the input * -1, if not, returns the input. sqrt, takes in a double and returns the square root of the double using the Babylonian method. sin, takes in a double between 0 and $2pi$ (if not throws an error) and returns the sin of it using taylor series. cos, takes in a double between 0 and $2pi$ (if not throws an error) and returns the cosine of it using taylor series. tan, takes in a double and uses the trig identity tan = sin/cos to computer the tangent and return it.

apply binary operator, takes in a binary operator and applies it to the top two elements in the global stack. apply unary operator, takes in a unary operator and applies it to the top two elements in the global stack. Both functions will return false if there is less than two elements in the stack, otherwise it will return true.





Results

Follow the instructions on the pdf to do this. In overleaf, you can drag an image straight into your source code to upload it. You can also look at `https://www.overleaf.com/learn/latex/Inserting_Images` **I found that my trigonemtric functions were the exact same as the output from libm's trigonemetric functions for the 10 digits after the decimal point I checked. Here is the code I used to write the outputs of each to a text file and graphed them using excel.**

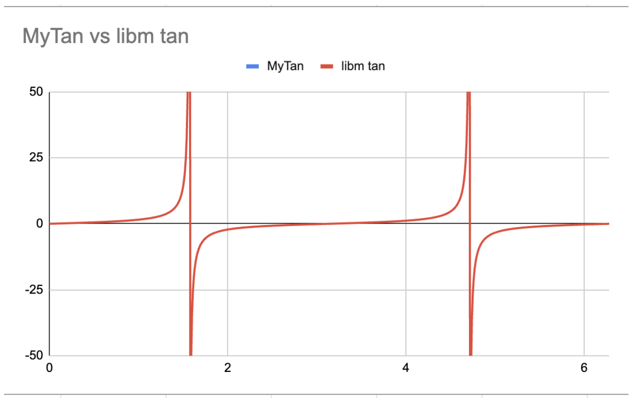
```
echo "" > tanOut.txt
echo "" > tan.txt

for i in $(seq 0.00 0.01 6.28)
do
   echo "$i t" | ./tests/calc_arm >> tanOut.txt
   echo "$i t" | ./calc >> tan.txt
done

exit 0
```

References

[1] Wikipedia contributors. C (programming language) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/C_(programming_language), 2023. [Online; accessed 20-April-2023].

[2] Robert Mecklenburg. *Managing Projects with GNU Make, 3rd ed.* O'Reilly, Cambridge, Mass., 2005.

[3] Walter R. Tschinkel. Just scoring points. *The Chronicle of Higher Education*, 53(32):B13, 2007.


MyTan vs libm tan
MyTan
libm tan
50
25
0
-25
-50
0
2
4
6