Assignment 4 – Calc Report Template

Jason Maheru

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.

- The purpose of this program is to function as a scientific calculator, using functions like cosine, sine, abs, sqrt, and tan functions.

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?
- What ways (other than changing epsilon) can we use to make our results more accurate? [1]
- What does it mean to normalize input? What would happen if you didn't?
- How would you handle the expression 321+? What should the output be? How can we make this a more understandable RPN expression?
- Does RPN need parenthesis? Why or why not?
- Yes there will be cases when sin and cosine formulas won't be able to be used. Such as invalid inputs and large-angle inputs probably. So either I convert the larger angles to smaller angles or produce an error.
- I believe I could use angle reduction using periodic identities, symmetry properties, angle addition, and subtraction formulas.
- Normalizing means adjusting the input of data to fit within a specific range for example angle values, 0 to 2pi or 0 degrees to 360 degrees. If you don't do that then it could lead to inaccurate results or loss of precision.
- I would handle "3 2 1 +" as 3 + 2 + 1 and make it equal to 6. To make it a more understandable RPN it would be "3 2 1 + +".
- RPN doesn't need parenthesis because RPN uses stacks to evaluate problems so "6 x (9 + 3)" is "6 9 3 + x".

[1] hint: Use trig identities

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.

1. Test for negative angles
2. Test for all errors
3. Test all the functions to see if they are working properly
4. Test for a basic test as "3 2 1 + +"
5. Test for all the -h
6. Test for cos, tan, sin, abs, sqrt

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[**?**][**?**][**?**].

- The calc program is used by first running the Makefile. You run the makefile by just inputting `make`. Then you run the program `./calc`. Now you can insert your calculation in RPN format. For example, if you want to do 1 + 1, you then enter `1 1 +` into the console. You can use absolute value, square root, sine, cosine, and tangent along with basic math along with fmod (`[s, c, t, a, r, %]` = [sin, cos, tan, abs, sqrt, fmod]). You can also call upon the -h flag `./calc -h` to get the help menu. Lastly to stop the program from running just do `^C` in the console also known as ctrl c.

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 in the last few assignments. Continue to answer it with the same level of detail and thought.

[3] This is my footnote.

- There are multiple files that the calc file calls upon. The calc file is where the input is taken and runs the error code checks and makes sure the input is valid. Calc then runs the computation based on what was entered by calling on the other files. The file mathlib.c is where all the abs, sqrt, sin, cos, and tan functions are defined. If any of those are inputted use the calc.c calls for them to do the calculation. The operators' file is where multiplication, division, subtraction, and addition are done since they are easy because all you have to do is pop the two values and solve. Then we have stack.c which is where the stack is pop, push, peek, clear, and print are defined.

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.

```
1. Define necessary data structures and constants:
   - Define STACK_CAPACITY for the size of the stack
   - Define a stack data structure witha size and array of doubles

2. Implement mathlib.c:
   - Implement Abs(double x) function:
     - Return the absolute value of x
   - Implement Sin(double x) function:
     - Normalize angle x to (0, 2)
     - Use the Maclaurin series to approximate sin(x)
   - Implement Cos(double x) function:
     - Normalize angle x to (0, 2)
     - Use the Maclaurin series to approximate cos(x)
   - Implement Tan(double x) function:
     - Normalize angle x to (0, 2)
     - Calculate tan(x) using sin(x) and cos(x) approximation

3. Implement operators.c:
   - Implement apply_binary_operator(binary_operator_fn op):
     - Pop two elements from the stack
     - Apply the binary operator op to the popped elements
     - Push the result back onto the stack
   - Implement apply_unary_operator(unary_operator_fn op):
     - Pop one element from the stack
     - Apply the unary operator op to the popped element
     - Push the result back onto the stack

4. Implement stack.c:
   - Implement stack_push(double item):
     - Push item onto the stack if capacity allows
   - Implement stack_peek(double *item):
     - Copy the top item of the stack to *item without popping
   - Implement stack_pop(double *item):
     - Pop the top item from the stack and store it in *item
   - Implement stack_clear():
     - Reset the stack size to zero
   - Implement stack_print():
     - Print all elements of the stack

5. Implement calc.c:
   - Implement main function:
     - Parse command-line arguments using getopt
     - Initialize the stack
     - Display prompt and read user input
     - Parse input and handle each token:
       - If token is a number, push it onto the stack
```

```
      - If token is an operator, apply it to the stack
    - Print the resulting stack after each operation
    - Handle errors and display help message if needed

6. Create tests.c:
   - Implement test cases for each function to ensure correctness and error handling
```

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; and 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.**

```
CALC.c
int main(int argc, char **argv){
    here is where the input is checked that if it is valid otherwise, provided error messages will
        be printed. Next, we will loop the input to see what operators are provided to ensure the
        right calculation is run. Also make sure valid number of inputs are provided.
}

MATHLIB.c
double Abs(double x) {
    evaluate absolute value
    return new value of x
}

double Sqrt(double x){
    find the sqrt of x
    return new value of x
}

double Sin(double x){
    find sin given x
    return value
}

double Cos(double x){
    find cos given x angle
    return value
}

double Tan(double x){
    Sin(x) / Cos(x)
    return value
}
```

```
OPERATORS.c
double operator_add(double lhs, double rhs){
    add the two values
    return val
}

double operator_sub(double lhs, double rhs){
    sub the two values
    return val
}

double operator_mul(double lhs, double rhs){
    mul the two values
    return val
}

double operator_div(double lhs, double rhs){
    divide the two values
    return val
}

STACK.c
bool stack_push(double item){
    check that the stack is not full
    else add the item to the stack
    return true
}

bool stack_peek(double *item){
    check that the stack is not empty
    else set item to the first item on top of the stack to show
    return true
}

bool stack_pop(double *item){
    check that the stack is not empty
    else set item to the the item on top of the stack
    delete the item on top since it is now set to *item
    return true
}

void stack_clear(void){
    set stack to 0
}

void stack_print(void){
    print the first item in the stack[0]
}
```

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`

-

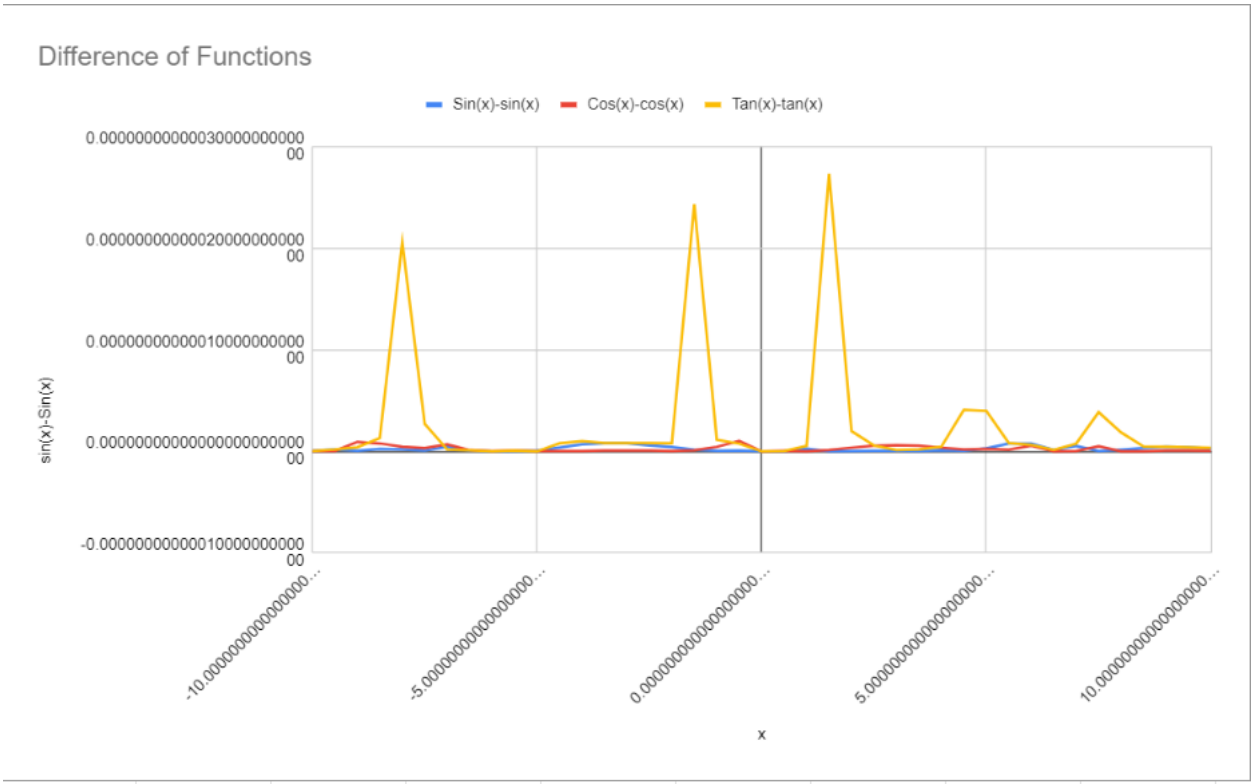

Figure 1: Enter Caption

References

- "How to Read a CSV File in C." YouTube, YouTube, 3 Aug. 2023, www.youtube.com/watch?v=X1Fq59EQaXo.
- Vanderbeek, Greg. "Order of Operations and RPN." DigitalCommons@University of Nebraska - Lincoln, digitalcommons.unl.edu/mathmidexppap/46/. Accessed 15 Feb. 2024.
- Kernighan, Brian Wilson, and Dennis M. Ritchie. The C Programming Language. Prentice-Hall, 1978.