Assignment 4 – Calc Report Template

William Zhou

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 perform various mathematical calculations. These include finding the absolute value, the square root, the sine, the cos, and then tan of a double. Of course, you can also perform the 4 core basic mathematical operations addition, subtraction, multiplication, and division. It can also mimic the stack ADT.

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?
  - You cannot use sin or cosine when you need a precise answer as the sin and cosine functions will create an approximation.
- What ways (other than changing epsilon) can we use to make our results more accurate? [1]
  - You can use trigonometric identities to simplify expressions and avoid unnecessary calculations that would create more errors.
- What does it mean to normalize input? What would happen if you didn't?
  - Normalizing input means insuring that input values stay in a consistent format within a specified range. If you did not normalize input, you could run into problems from inputs of various forms. Ex. inputting angle in radians and then degrees
- How would you handle the expression 321+? What should the output be? How can we make this a more understandable RPN expression?
  - I would output an error message as the intent of the user is unclear. You can make this a more understandable RPN expression by actually formatting this in correct RPN expression. Does the user want to add 32 with 1? Then seperate them with a space. Do they want to add 3,2,1 all together? Then write it as 3 2 + 1 +.
- Does RPN need parenthesis? Why or why not?
  - No. You do not need parenthesis as the order of calculations as it calculates from left to right regardless of the hierarchy of operations from traditional formats.

[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 that mathematical function outputs match expected outputs 2. Test to make sure all errors print to stderr 3. Test to make sure error messages match expected error messages 4. Test function inputs are of valid typing and value 5. Test outputs from arithmetic functions match or approximate expected value 6. Test that stack ADT functions match expected output and do not overfill the stack

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].

First, make sure you type out "make" to compile the code and create an executable. Afterwards, run ./calc followed by your desired mathematical function. There are several available to you.

Mathlib functions:

Abs(double x): Takes in a double and returns the double in absolute value

Sqrt(double x): Takes in a double and returns the square root

Sin(double x): Takes in a double and returns the sin value of it

Cos(double x): Takes in a double and returns the cos value of it

Tan(double x): Takes in a double and returns the tan value of it

Operator functions:

apply_binary_operator(binary_operator_fn op): Takes in an operator and applies that operation to the top 2 elements in the stack. The resulting value is then placed on top of the stack.

apply_unary_operator(unary_operator_fn op): Takes in an operator and applies that operation to the first element on the stack. The result is then pushed to the stack. If there are not enough elements to pop within the stack, an error will be returned.

operator_add(double lhs, double rhs): Takes in 2 doubles, adds them, and returns the result.

operator_sub(double lhs, double rhs): Takes in 2 doubles, subtracts the rhs from the lhs, and returns the result.

operator_mul(double lhs, double rhs): Takes in 2 doubles, multiplies them, and returns the result.

operator_div(double lhs, double rhs): Takes in 2 doubles, divides the lhs by the rhs, and returns the value.

parse_double(const char *s, double *d): Parses double-precision floating number from string s and stores the location in d, then returns true. If string is not valid, returns false.

Stack functions:

[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.

stack_push(double item): Pushes item to the top of the stack, and updates stack_size. If successful, returns true. Otherwise, if stack is full, returns false and the stack is unchanged.

stack_peek(double *item): Copies the first item of the stack to the address pointed to by item, then returns true. If stack was empty, returns false.

stack_pop(double *item): Stores the first item of the stack in memory location pointed to by item, then returns true. Returns false if stack is empty.

stack_clear(void): Sets the size of the stack to zero.

stack_print(void): Prints every element in the stack to stdout.

Have fun performing calculations!

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.

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.

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.**

The calculator functions are seperated into 3 files. These files are mathlib.h, operators.h, and stack.h. The absolute value, square root, sin, and cosine functinos are in mathlib.h. The mathematical operationi functions such as add, subtraction, multiply, and divide are in operators.h. Stack.h contains the stack functions as the name would imply.

Here is a more indepth look at the functionality of each function within the 3 files.

Mathlib functions:

Abs(double x): Takes in a double and returns the double in absolute value

Sqrt(double x): Takes in a double and returns the square root

Sin(double x): Takes in a double and returns the sin value of it

Cos(double x): Takes in a double and returns the cos value of it

Tan(double x): Takes in a double and returns the tan value of it

Operator functions:

apply_binary_operator(binary_operator_fn op): Takes in an operator and applies that operation to the top 2 elements in the stack. The resulting value is then placed on top of the stack.

apply_unary_operator(unary_operator_fn op): Takes in an operator and applies that operation to the first element on the stack. The result is then pushed to the stack. If there are not enough elements to pop within the stack, an error will be returned.

operator_add(double lhs, double rhs): Takes in 2 doubles, adds them, and returns the result.

operator_sub(double lhs, double rhs): Takes in 2 doubles, subtracts the rhs from the lhs, and returns the result.

operator_mul(double lhs, double rhs): Takes in 2 doubles, multiplies them, and returns the result.

operator_div(double lhs, double rhs): Takes in 2 doubles, divides the lhs by the rhs, and returns the value.

parse_double(const char *s, double *d): Parses double-precision floating number from string s and stores the location in d, then returns true. If string is not valid, returns false.

Stack functions:

stack_push(double item): Pushes item to the top of the stack, and updates stack_size. If successful, returns true. Otherwise, if stack is full, returns false and the stack is unchanged.

stack_peek(double *item): Copies the first item of the stack to the address pointed to by item, then returns true. If stack was empty, returns false.

stack_pop(double *item): Stores the first item of the stack in memory location pointed to by item, then returns true. Returns false if stack is empty.

stack_clear(void): Sets the size of the stack to zero.

stack_print(void): Prints every element in the stack to stdout.

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`

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.