Vincent Siu
10/6/22
Init Design

Final Design Assignment 2

Start code for mathlib.c

Import the math.h library
Import mathlib header file.

Set global variable epsilon which will equal 1^-10

#write helper abs/ power function
Declare the function as type double and input being type double
    If x is larger than 0, return x, otherwise return x * -1.

#Code for square root goes here

#Code for E(x) goes here

Declare epsilon, which should equal 1/1000000000

#my_log function
Declare the mylog function with the return type being double and the argument being “x”.
Set the previous value (x_0) equal to 1.0

Call Exp() with the previous value being the input.

While absolute value of (Exp(x) - x is > epsilon)
    Set a variable exp_x equal to the output of Exp(previous value)
    To calculate current, add previous value to (x/exp_x)/exp_x
    Set previous equal to current value.
Return current value

#my_sin function
Declare the my_sin function with the return type being double and the argument being ‘x’.

For sin to work with negative numbers, we are going to make x positive using the abs_val function, or |x|.
Set a variable to track iterations = 1.

Vincent Siu
10/6/22
Init Design

We are going to calculate our first iteration: current number = ||x|/2*iterator| * ||x|/2*iterator+1| * |x|

While the absolute value of the current term is > epsilon:
    Set |current| equal to ||x|/2*iterator| * ||x|/2*iterator+1| * |previous|
    If the iterator number is odd, set the current value to be negative.
    Add the current var to the sin taylor expansion sequence
    Set the variable previous equal to current
    End loop
If x is negative, negate the sequence, otherwise return the sequence as is.

#my_cosine function
Call sin function with arguments being pi/2 - mycos input
Return the result.

#function for my_arcsin

Initialize the previous value to be equal to 0.

To calculate our first current value, previous value - (sin(previous value)-x)/cos(previous value)

While the absolute value of the difference between current and previous is > epsilon
    Set the previous value equal to the current value
    Calculate the current, using that same equation from above.
Return the current value.

#code for my_arccos
Declare the my arc-sin function with the return type being double with parameter being x
Return (pi / 2-sin^-1(x))

#my_arctan function
Declare function for arctan x with return type double and parameter x
return (sin^-1(x/square root(x^2 + 1)))

Vincent Siu
10/6/22
Init Design

Start code for main (mathlib-test.c)

Import stdio
Import c math library
Import mathlib header file
Import unistd header file
Define the choices to run this function to be a,s,c,S,C,T, and l.

Declare the main function
Using a while loop, get character input with getopt() with options being a,s,c,S,C,T,l

Make variables to track down the number of tests run for each function, and set all of them equal to 0.

Using switch statements
If the case is equal to -a:
If the test has not been run (or number of times test run is equal 0)
For each i from 0 to 2pi(all inclusive), incrementing by 0.05pi each
Call the my sin function with the argument being the iterator:
Call the sin function from mathlib
Print the results in this order: argument, result of my sin function, result from library, difference
If this test has not been run
For each i from 0 to 2pi(all inclusive), incrementing by 0.05pi each
Call the my cos function with the argument being the iterator:
Call the cos function from mathlib
Print the results in this order: argument, result of my cos function, result from library, difference

If this test has not been run
For each i from -1(inclusive) to 1, incrementing by 0.05 each
Call the my arcsin function with the argument being the iterator:
Call the arcsin function from mathlib
Print the results in this order: argument, result of my arcsin function, result from library, difference

If this test has not been run
For each i from -1(inclusive) to 1, incrementing by 0.05 each
Call the my arccos function with the argument being the iterator:

Vincent Siu
10/6/22
Init Design

Call the arccos function from mathlib
Print the results in this order: argument, result of my arccos function, result from library, difference

If this test has not been run
For each i from 1(inclusive) to 10, incrementing by 0.05 each
Call the my my log function with the argument being the iterator:
Call the log function from mathlib
Print the results in this order: argument, result of my log function, result from library, difference

If this test has not been run
For each i from 1(inclusive) to 10, incrementing by 0.05 each
Call the my arctan function with the argument being the iterator:
Call the tarctan function from mathlib
Print the results in this order: argument, result of my arctan function, result from library, difference

If the case is equal to -s
If this test has not been run
For each i from 0 to 2pi, incrementing by 0.05pi each
Call the my sin function with the argument being the iterator:
Call the sin function from mathlib
Print the results in this order: argument, result of my sin function, result from library, difference
If the case is equal to -c
If this test has not been run
For each i from 0 to 2pi, incrementing by 0.05pi each
Call the my cos function with the argument being the iterator:
Call the cos function from mathlib
Print the results in this order: argument, result of my cos function, result from library, difference
If the case is equal to -S
If this test has not been run\
For each i from -1(inclusive) to 1, incrementing by 0.05 each
Call the my arcsin function with the argument being the iterator:
Call the arcsin function from mathlib
Print the results in this order: argument, result of my arcsin , result of library, difference
If the case is equal to -C
If this test has not been run
For each i from -1(inclusive) to 1, incrementing by 0.05 each
Call the my arccos function with the argument being the iterator:
Call the arccos function from mathlib

Vincent Siu
10/6/22
Init Design

Print the results in this order: argument, result of my arccos, result from library, difference

If the case is equal to -T

If this test has not been run

For each i from 1(inclusive) to 10, incrementing by 0.05 each

Call the my arctan function with the argument being the iterator:

Call the arctan function from mathlib

Print the results in this order: argument, result of my arctan function, result from library, difference

If the case equal to -l

If this test has not been run

For each i from 1(inclusive) to 10, incrementing by 0.05 each

Call the my my log function with the argument being the iterator:

Call the log function from mathlib

Print the results in this order: argument, result of my log function, result from library, difference

Vincent Siu
10/6/22
Init Design