Isaac Flores
Prof. Miller
CSE13S
October 7, 2022

General Idea:

Mathlib.c:

my_sin(x): Get the Maclaurin series of sin(x) to approximate the value of sin(x) from the domain of 0 to 2pi in steps of 0.05pi.

my_cos(x): Get the Maclaurin series of cos(x) to approximate the value of cos(x) from the domain of 0 to 2pi in steps of 0.05pi.

my_arcsin(x): Get the Maclaurin series of arcsin(x) to approximate the value of arcsin(x) from the domain of -1 to 1 in steps of 0.05.

my_arccos(x): Subtract pi/2 by the approximation of arcsin(x) from the previous function to get arccos(x).

my_arctan(x): Use a simple evaluation of arctan(x) which gets arcsin(x / square root of (x^2 +1)). This evaluation will be equal to arctan(x).

my_log(x): Use the Newton-Raphson method to approximate ln(x) which gets the inverse of e^x. This method gets the previous value and adds it to ((x - e^(previous value))/ e^(previous value)).

Helper Functions:

factorial(n): Iterate from n down to 1 by steps of -1. Set the previous value to itself times n. Accumulate this value and return it when n = 1. Return this value.

my_pow(x, y): Iterate from y down to 1 by steps of -1. Accumulatively multiply x by itself every iteration until y = 1. Return this value.

Mathlib-test.c:

Import the functions from mathlib.c. Check the options that the user inputted when running mathlib-test. Print the header lines and then loop through the respective domain of the function. Print the x value, my approximation of the respective function, the math.h version of the respective function, and the difference between the two.

Pseudocode:

Mathlib.c:

my_sin(x):

set n =1, answer = x, and previous value = x
do
    # increment n by 2
    # set the current value to -1 * previous value * (x^2 / n(n-1)). Set the previous value to the current value.
    # set the answer equal to itself plus the current value.
    # get the absolute value of current.
while the absolute value of the current value is > 10 ^ -10, keep iterating or else return the answer.

my_cos(x):

set n =0, answer = 1, and previous value = 1
do
    # increment n by 2
    # set the current value to -1 * previous value * (x^2 / n(n-1)). Set the previous value to the current value.
    # set the answer equal to itself plus the current value.
    # get the absolute value of current.
while the absolute value of the current value is > 10 ^ -10, keep iterating or else return the answer.

my_arcsin(x):

set previous value = x
do
    # set the current value to the previous value - ((sin(previous value) - x) / cos(previous value)). Set a compare variable to the previous value. Set the previous value to the current value.
    # get the absolute value of the previous value which is now the current value minus the compare variable which is now the previous value.
while the absolute value of the previous equation is > 10 ^ -10, keep iterating or else return the previous value which is now the current value.

my_arccos(x):

subtract pi/2 by arcsin(x) made in the previous function
return that value

my_arctan(x):

get the value of arcsin(x / square root of (x^2 + 1))

return that value

my_log(x):
set previous value = 1
do
    # set the current value to the previous value + ((x - e^(previous value))/ e^(previous value))
    # set the previous value to the current value
while (e^(current value) - x) > 10 ^ -10, keep iterating or else return the current value

<u>Helper Functions:</u>
factorial(n):
set the start value and previous value to n.
if n = 0 return 1
else
    # iterate the start by steps of -1
    # set the previous value to multiply itself by the start value.
    # if the start value = 0 end the iterations and return the previous value

my_pow(x, y):
set the previous value to 1 and the current value to 0
set the power to y
iterate the power by steps of -1
    # set the current value equal to the previous value times x.
    # set the previous value to the current value
end the iteration when the power = 0 and then return the previous value.

<u>Mathlib-test.c:</u>
initilize counter variables for each function call and set them equal to zero
check the options the user inputted after calling mathlib-test and stop when all options have been read
if the respective letter for a function is called and the counter variable is equal to 0
    # print the two header lines being x, respective function, library, difference, and the dashes '-'.
    # for loop through the respective function's domain [0,2pi] for sin and cos, [-1, 1] for arcsin and arccos, and [1, 10] for arctan and log. Increment by .05 for arcsin, arccos, tan, and log and 0.05pi for sin and cos.
    # on every iteration print the x value, my version of the respective function with x plugged in, the math.h version of the function with x plugged in, and the difference between the two.

    # add 1 to the respective counter variable to ensure repeated calls aren’t shown
if ‘a’ is inputted print the tables for all the functions and exit the loop.