Edison Chen

Assignment 2: A Little Slice of Pi

Program Description:

This program implements a number of mathematical functions similar to those found in the math.h library. The functions will implement different ways of computing fundamental constants e and π. The program will implement a dedicated test harness, mathlib-test.c for comparing the results of the functions with that of those in the math.h library. Each of the functions will halt computation upon reaching a threshold $\varepsilon = 10^{-14}$ defined in the given mathlib header file.

Files Included:

- bbp.c - contains functions pi_bbp(c) and pi_bbp_terms(). pi_bbp() will approximate π using the Bailey-Borwein-Plouffe formula. pi_bbp_terms() will track the number of computed terms.
- e.c - contains functions e(c) and e_terms(). e() will approximate the value of e using Taylor series. e_terms() tracks the number of computed terms.
- euler.c - contains functions pi_euler() and pi_euler_terms(). pi_euler() will approximate π using Euler's solution to Basel. pi_euler_terms() will track the number of computed terms.
- madhava.c - contains functions pi_madhava() and pi_madhava_terms(). pi_madhava() will approximate the value of π using the Madhava series. pi_madhava_terms() will track the number of computed terms.
- Mathlib-test.c - test harness that compares values of implemented functions with functions from the math.h library.

- newton.c - contains functions sqrt_newton() and sqrt_newton_iters(). sqrt_newton() will approximate the square root of the argument passed to it using the Newton_Raphson method. sqrt_newtons_iters() will track the number of iterations taken.
- viete.c - contains functions pi_viete() and pi_viete_factors(). pi_viete() will approximate π using Viete's formula. pi_viete_factors() will return the number of computed factors.
- mathlib.h - header file used for the interface of the implemented math library.
- Makefile - file that builds the program, runs the program, and formats all files to clang format.
- README.md - text file in markdown format that describes: the program, how to build the program, and how to run the program.
- DESIGN.pdf - pdf that describes the design and program
- WRITEUP.pdf - pdf containing explanations for any errors and graphs comparing values of implemented math functions and math functions in math.h library.

Pseudocode:

*each of these files contains a function to count the number of iterations/computations

e.c:

for each iteration of the function

    factorial = factorial * iteration_count

    next_value = 1/(factorial)

    total = total + next_value

when next_value is less than epsilon, end the loop

return total

from assignment 2 doc, **newton.c:**

function takes an argument, x

z = 0.0

y = 1.0

while the absolute value of y-z is greater than epsilon

    set z = y

    y = 0.5 times (z + x / z)

return y

bbp.c:

for each iteration of the function

    for each number i up to k

        next_value = next_value * 1/(16 * i)

        total = total + (next_value * (4/(8i+1) - 2(8i+4) - 1/(8i+5) - 1/(8i+6)))

    if next_value < epsilon

    stop looping and return total

euler.c:

for each iteration of the function

    next_value = 1/(iteration count * iteration count)

    total = total + next_value

    when next_value < epsilon

        multiply total by 6

        square root total using sqrt function in newton.c

        return total

madhava.c:

for each iteration of the function

    for each number i up to k

        next_value = next_value * 1/(-3 * i)

        total = total + next_value * 1/(2k+1)

    when next_value < epsilon

    multiply total by square root of 12, using sqrt function in newton.c

viete.c:

for each iteration of the function

for all k > 1

    next_value = sqrt function in newton.c to take the square root of (2 + next_value)

    total += next_value

    when next_value < epsilon

        total = inverse of total and multiply by 2

        return total

from assignment 2 doc, **mathlib-test.c:**

define OPTIONS aebmrvnsh

while command line option in OPTIONS

switch cases for each command a, e, b, etc.

each of the implemented math functions will have a print statement comparing the output of the implemented function with the function from the math.h library and display the difference between the two to 15 decimal places

Notes about pseudocode:

- many of the functions above have not been implemented yet, but more details and notes will be added once implemented
- many of the functions have the same structure, where a for loop can be implemented in place of a sigma and exponents can be written out as 2^(k) = 2^(k-1) * 2
- Each of the mathematical c files also have a function that count the number of terms/iterations until the loop ends

Credit:

- I used the newton.c sqrt function provided by Professor Long in the assignment 2 pdf
- The getopt() parsing example in the assignment 2 pdf was used as a reference when writing my mathlib-test.c file
- I attended Eugene's section on 10/5/21 for general guidance and instruction for assignment 2