Assignment 2 design

Description of Program

This program consists of 8 files that, together, contain various methods of calculating mathematical constants such as e and pi. This program will not implement the constants from math.h, but will compare the final values calculated with the math library. The various methods will calculate this while the difference between the previous and next term of the summation are greater than epsilon, which is a constant defined in the given mathlib.h file. Since the various methods outlined to calculate these constants are in the form of summations, I will use for or while loops to implement these calculations.

Files included:

- mathlib-test.c
  - This is the testing harness for the functions that will approximate the constants e and pi.
- e.c
  - This file will contain two functions, e() and e_terms(). The first of which will approximate the value of e with the taylor series $e = \sum_{k=0}^{\infty} (1/k!)$. The second will return the number of iterations
- madhava.c
  - This file will contain two functions, pi_madhava() and pi_madhava_terms(). pi_madhava() will compute pi using the summation listed $\pi/\sqrt{12} = \sum_{k=0}^{\infty} (-3)^{-k}/(2k+1)$. pi_madhava_terms() will return the number of iterations used for the approximation
- euler.c
  - This file will contain two functions, pi_euler() and pi_euler_terms(). pi_euler() will compute pi using the summation listed $\pi = \sum_{k=1}^{\infty} \frac{1}{k^2}$. pi_euler_terms() will return the number of iterations used for the approximation
- bbp.c
  - This file will contain two functions, pi_bbp() and pi_bbp_terms(). bbp_euler() will compute pi using the summation listed $\pi \sum_{k=0}^{n} = 16^{-k}(\frac{(k(120k+151)+47}{k(k(k(512k+1024)+712)+194)+15})$. pi_bbp_terms() will return the number of iterations used for the approximation
- viete.c
  - This file will contain two functions, pi_viete() and viete_factors(). pi_viete() will compute pi using the summation listed $\frac{2}{\pi} = \prod_{k=1}^{\infty} \frac{a_i}{2}$. viete_factors() will return the number of iterations used for the approximation.
- newton.c

    - This file will contain two functions, sqrt_newton() and sqrt_newton_iters(). sqrt_newton() will compute the square root using the equation listed $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$. sqrt_newton_iters() will return the number of iterations used for the approximation.
- mathlib.h
    - This is a given header file with function declarations.
    - This is pulled from the resources directory supplied by the professor
- Makefile
    - This will contain all the code for building the program using clang
    - Will include make clean for removing all compiler generated files
    - Will link all the files to mathlib executable
- README.md
    - This is an explanation of proper programs usage and command line arguments written in markdown syntax
- Design.pdf
    - This is the document being currently viewed, it contains pseudocode explanations and description of the program.
- Writeup.pdf
    - This will contain a writeup of graphs displaying the difference between values reported by function and math libraries
    - Analysis and explanations for discrepancies and findings

Pseudocode:

mathlib-test.c

- uses getopt to parse command line inputs, the choices of inputs are a, e, b, m, r, v, n, s, h.
- Uses boolean values to check which command line arguments are received
    - While getopt is not -1
        - If the command line argument is a: run all tests.
        - If the command line argument is e run the test for e.
        - If the command line argument is b run the bbp pi test.
        - If the command line argument is m run the Madhava pi test.
        - If the command line argument is r run the Euler test for pi.
        - If the command line argument is v run Viete's test for pi.
        - If the command line argument is e run the newton square root tests in a range from 0 to 10 incrementing by 0.1.
        - If the command line argument is s display term count for whatever test was run.
        - If the command line argument is h display usage information. Also do not run any tests

e.c

- e()
    - Begin iteration at the first term of the above listed summation.
        - This means that the output and the term start at 1.
    - While the value of the term added is greater than the given EPSILON value, the next term is the previous term divided by the number of the iteration

        - Add the term to the sum
    - Return the sum.
- e_terms()
    - Return the number of terms computed.

madhava.c

- pi_madhava()
    - Initialise the sum, term and counter since you start at the first term.
    - While the absolute value of the term is greater than epsilon.
        - Set the term equal to the previous term multiplied by -3 and (2(k-1) +1)/(2k+1) when k is the current counter value
        - Add the term to the sum.
        - Add one to the counter.
    - Multiply the sum by the square root of 12.
    - Return the sum.
- pi_madhava_terms()
    - Return the counter value.

newton.c

- sqrt_newton():
    - takes one input of the number to compute square root of
    - Initialize previous guess and counter as zero.
    - Initialize current guess as one.
    - While the absolute value of this guess minus the previous is greater than epsilon.
        - Add one to the counter.
        - Set the guess value equal to the previous value.
        - Set the guess value to half of the previous guess plus the input divided by previous guess.
    - Return the guess.
- sqrt_newton_iters()
    - Returns number of iterations computed.

bbp.c

- pi_bbp()
    - Set the counter value to 1.
    - Set the sum to 47/15 and the first term to 47/15 since these are the 1st terms of the summation.
    - While the term is greater than epsilon.
        - The current term is equal to the previous term time 16 times (k(120k+151)+47) time k-1(k-1(k-1(512k+1024)+712)+194)+15 divided by k(k(k(512k+1024)+712)+194)+15 and divided by (k-1(120k-1+151)+47). Where k is the counter value.
        - Increment the counter by one
        - Add the term to the sum.
    - Return the sum.
- pi_bbp_terms()
    - Return the counter value.

viete.c

- pi_viete()
    - Start the counter value at zero.
    - Initialize the output as 1.
    - Initialize the first guess as the 0.
    - While the absolute value of 2/(guess - 1) is greater than epsilon.
        - The guess is equal to the square root of 2 plus the previous guess
        - Multiply the output by 2/guess.
        - Increment the counter by one.
    - Multiply the output by 2
- pi_viete_factors()
    - Return the counter value.

Notes

- mathlib-test.c: This file has one function, main(), which takes two inputs, int argc and char **argv.
    - This file contains the test harness for the rest of the functions
    - This file runs the rest of the functions and compares them with mathlib values.
- e.c:
    - This file contains the functions e() and e_terms().
    - This file computes e until the step is less than epsilon, which is given in the mathlib function
    - e_terms returns the amount of times the function has iterated
    - This uses the Taylor series expansion of e^x
- sqrt.c:
    - This file uses newton's method of approximating the square root.
    - This file is adapted from the python code in the assignment document.
- viete.c
    - This function inverts the repeated multiplication, so that instead of computing ai/2 where ak is sqrt(2+ak-1) , it computes that 2*2/ai where ai=sqrt(2k-1).
    - The term of multiplication converges on 1, thus term - 1 is what the epsilon value is computed off of.
- madhava.c
    - This calculates from the first term, since the way to check the next term is off the previous term.
- bbp.c
    - This calculates from the first term, since the way to check the next term is off the previous term.
- euler.c
    - This function converges very slowly.