Assignment 2 DESIGN.pdf

Description of Program

This program simulates calculating mathematical constants e and pi. These constants are calculated term by term from summations and terms required are determined by once the error between terms to become less than the given constant EPSILON e $=10^{-14}$. The main program mathlib-test.c compiles all the methods of solving e and pi and allows for the users to use created commands denoted with a '-' that print out statistics and differences between the methods. The commands'-h' show how to use the program along with all the possible commands the program can run. Overall the program uses multiple methods of calculating pi and can reveal data such as the amount of terms required to reach pi or e allowing for the user to better understand how well the methods can approximate pi.
Files to be included in the "asgn1" files

e.c:
Implements e() and e_terms()

bbp.c:
Implements pi_bbp() and pi_bbp_terms()

euler.c:
Implements pi_euler() and pi_euler_terms()
madhava.c:
Implements pi_madhava() and pi_madhava_terms()

mathlib-test.c:
This contains the main() function which tests each of your math library functions.
Contains the commands
-a : Runs all tests.
-e : Runs e approximation test.
-b : Runs Bailey-Borwein-Plouffe π approximation test.
-m : Runs Madhava π approximation test.
-r : Runs Euler sequence π approximation test.
-v : Runs Viète π approximation test.
-n : Runs Newton-Raphson square root approximation tests.
-s : Enable printing of statistics to see computed terms and factors for each tested function.

-h : Display a help message detailing program usage.

mathlib.h:
This contains the interface for your math library.

newton.c: Implementation of the square root approximation sqrt_newton()and the function to return the number of computed iterations sqrt_newton_iters().

viete.c:
This contains the implementation of Viète's formula to approximate π and the function to return the number of computed factors. Includes pi_viete() and pi_viete_factors()

Makefile:
CC = clang must be specified.
CFLAGS = -Wall -Wextra -Werror -Wpedantic must be specified.
make must build the mathlib-test executable, as should make all and make mathlib-test.
make clean must remove all files that are compiler generated.
make format should format all your source code, including the header files

Linked source file to main through command prompt and allowing for new source files to be run quickly. It also cleans up the file format tools and makes it more accessible

DESIGN.pdf:
Describes the entire assignment along with all information regarding the program

README.md
Describes how to build and run the program

WRITEUP.pdf
Includes graphs comparing the methods and terms used to approach e and pi. Also uses

Source code for e.c:
Includes mathlib.h
Initialize Double error = 1
Initialize Double term = 0
Initialize Double e = 1
Initialize Double preterm =1
Initialize Double pree

Initialize cont =1

While (error>EPSILON(defines as 10^{-14}))
    pree=e
    E is the summation of (cont/(term*preterm))
    Preterm = term *preterm keeps track of the factorial
    Term is keeping track for the looping
    Error= e-pree
Return out e and term

Source code for euler.c:
Includes mathlib.h
Initialize int eulerterms =0
Create function double pi_euler
    Create double variables for error,prepi,pi,pow
    Ensure that counter eulerterms is reset to 0;
For k =1 such that EPSILON <error k interates by 1
    Pow equals 1/(k*k) according to eulers method is the functioning term of the summation
    Prepi set to pi
    Pow is added to pi
    Error calculator the difference between the terms
    Eulerterms count +1
Pi outside of the summation is sqrt_newton (pi*6) which finalizes the method
Return pi

Int terms counter function pi_euler_terms()
    return eulerterms

Source code madhava.c:
Includes mathlib.h

Initializes terms as 1
Creates function double pi_madhava()
    Double error set as 1
    Double prepi,pi =1 ,pow =1
    Counter terms set to 1
    For k =1 as Epsilon <error k interates by 1

For i =1 i<=k i interates 1
Pow is multiples to -⅓
Prepi equals pow/(2*k+1)
Pi adds prepi
Implement absolute value of prepi using if and else if checking for negatives then reversing to positive
Pow is reset to 1
Terms counter is added +1
Pi equals sqrt_newton() using newton.c implementation * pi
Return pi
Create int term function pi_madhava_terms()
Returns terms

Source code bbp.c

Int bbpterms initialized 1

Create double pi_bbp()

Double error=1
Double prepi
Double pi (47/15)
Double pow =1
Double base
Restate bbpterms=1
For k =1 as Epsilon <error k interates by 1
For i =1 i<=k i interates 1
Pow is multiples to 1/16
prepi = pi
Base = (k * (120.0 * k + 151.0) + 47.0) / (k * (k * (k * (512.0 * k + 1024.0) + 712.0) + 194.0) + 15.0)
pi adds pow * base
error between pi and prepi
pow reset to 1
Return pi
Create function int pi_bbp_terms()
Return bbpterms

Source code newton.c

Initialize fabsminus which is absolute value function for floats
sqrterms=0
Declare double sqrt_newton(double x)
    Double z=0
    Double y=1
    sqrterms=0
    While (fabsminus(y,z) < EPSILON){
        z=y
        y= .0*(z+x/z)
        Sqrterms +=1;
    Return y
Absolute value function double fabsminus(double y, double z)
 double zero = 0;
if (y - z > zero)
    return y - z
else
    return (-1 * (y - z))

Count interactions int sqrt_newton_iters()
 return sqrterms

Source code viete.c:
Include mathlib.h
Declare vieterms =0

Function double pi_viete()
    double error =1
    double prepi
    double pi = base case which is sqrt_newton(2)/2.0
    Double pow = sqrt_newton(2) which is a constant that will be reused
    Vieterms restated as 0
    While (EPSILON < error)
        pow=sqrt_newton(pow_2)
        Prepi set equal to pi
        Pi multiply by pow/2
        Error between prepi and pi

Vieterms add 1
pi=2/pi
Int function pi_viete_factors()
Return veiterms

Source code mathlib-test.c
include mathlib.h
include math.h
include stdbool.h
include stdio.h
include unistd.d
define OPTIONS "aebmrvnsh"

In main (int argc, char) taken from Prof. Long
Boolean of all the commands corresponding to Options
While (opt =getopt(argc,argv,OPTIONS))!=1)
Switch opt
Case for each boolean command and set to true then break
For case a however no boolean corresponding set et,m,r,b,v,n to true
Case h help will print designated message and not trigger any other commands

All the other booleans will set off prints statements according to each commands setting up (function = function , M_E/M_PI =,diff=fabs(function-M_E/M_PI)
Case s triggers term print statement under each print statement
Return 0

Source code notes:
Int term:Each source file except for mathlib-test had an term counter variant that performs the same task
double e: retain the summation of the method to calculate e
double pree: retains the previous summation of the terms of e.
double pi: retain the summation of the method to calculate pi
double prepi: retains the previous summation of the terms of pi.
double pow/base: was the variable that would apply some sort of mathematical operation to pi or e
double error: takes the abs of pi and prepi/e and pree to compare against EPSILON
Took and understand of sqrt_newton from asgn2 doc

Took the basics of Makefile from Euguene
Took basics of mathlib-test.c from asgn2 doc