Assignment 2 DESIGN.pdf

Vincent Liu
1/29/23

Purpose

The purpose of this assignment is to emulate the <math.h> library that is used with the C programming language and compute some functions in a similar manner to the library, all without using the library itself. The output of each function written should also be tested for accuracy against outputs of the math library. This is done to exhibit how the student written program works and is accurate. The assignment requires students to estimate the value of *e* (Euler's number", the value of pi, and square roots.

Description:

1. e.c
    - Contains two functions that approximates the value of 'e' (Euler's number) & track the number of terms computed, as well as return the number of computed terms
2. madhava.c
    - Contains two functions that approximate the value of pi using the Madhava series and track the number of computed terms. The other function returns the number of computed terms
3. euler.c
    - Contains two functions that approximate the value of pi using Euler's solution and track the number of computed terms. The other function returns the number of computed terms
4. bbp.c
    - Contains two functions. The first function approximates the value of pi using the Bailey-Borwein-Plouffe formula and tracks the number of computed terms. The other function returns the number of computed terms
5. viete.c
    - Contains two functions. The first approximates the value of pi using the Viete's formula and tracks the number of computed terms. The other function returns the total number of computed terms
6. newton.c
    - Contains two functions. The first approximates the square root of an input using the Newton-Raphson method. This function adds to the number of iterations. The

second function will return the total number of iterations taken in determining the square root of a number.

7. mathlib-test.c
    - Contains the main() function used to call/execute the various student programmed functions depending on the argument input from the terminal.
8. mathlib.h
    - Includes the definition of EPSILON that is used in all .c files for this assignment. Also include the function ‘absolute’ that is similar to the ‘abs’ function found in math.h.
9. Makefile
    - Used to clean directory and generate associated executables
10. README.md
    - Text file in Markdown format that describes how to build and run the program.
11. DESIGN.pdf
    - Describes the design for the program thoroughly with pseudocode and visualizations
12. WRITEUP.pdf
    - Describes what the program does, and gives insight on the results found about the method's efficiency as well as other details.

Structure of Program

Main file:

- mathlib-test.c

Files called by mathlib-test.c:

- e.c
- bbp.c
- madhava.c
- euler.c
- Viete.c
- newton.c

mathlib.h is used by all files.

Pseudocode

e.c:

global variable counter

E function:

Set counter to unsigned int
Initialize double variables
Set value of denominator and counter to 1
While difference between new result and old result is greater than EPSILON:
    Set prev result to current one
    result+=1/denominator
    Denom equal denom*counter
    Add 1 to counter
Subtract 1 from counter
Return the result

e_terms:

    Return counter

Madhava.c

Pi_madhava:

Initialize double variables and int variable for counter
Make a loop (condition checks if 1/denominator is greater than epsilon)
    Every loop adds to counter
    Calculate denominator
    Update output variable
    Update power to be multiplied by in next iteration
Return sqrt(12)*output

pi_madhava_terms:

    Return computed terms

Euler.c

Pi_euler
Initialize variables
Initialize counter variable
While loop with condition: new fractions calculated is larger than epsilon)
    New calculated fraction will be 1/counter * counter
    Accumulate fractions
    Add to counter
Subtract from counter (since it was set to 1 before any iteration)
Return sqrt(6*cumulative fraction(s))

Pi_euler_terms
    Return computed terms

Bbp.c
pi_bbp
Set counter as an unsigned integer
Initialize variables
While loop with condition: new calculated fraction is greater than epsilon
    Calculate numerator and denominator
    Update the output
    Save and update previous iteration
    Add to counter
    Update power
Return output

pi_bbp_terms
    Return computed terms

Viete.c
Pi_viete
Initialize counter as unsigned integer
Initialize variables as double
While loop with condition: absolute(difference between iterations values) is greater than epsilon
    Calculate numerator
    Update previous numerator
    Update output

Add to counter
Subtract from counter (counter started at 1 on iteration 0)
Return 2/output (based on formula)

pi_viete_factors

Return counter

Newton.c

sqrt_newton

Initialize counter as global variable
Initialize two variables
While absolute value of difference of two variables is greater than epsilon
guess=output (previous iteration)
b = 0.5 * (guess + input / guess)
Add to iteration
Return square root output

sqrt_newton_iters

Return iterations

---

mathlib-test.c

Include mathlib.h, errno.h, math.h, stdbool, stdio.h, stdlib.h, unistd.h

Function to print if there is input error
Main function detects argument/character inputs with use of getopt
Use booleans to decide what to print
If statements to decide what should be printed based on input (if -s is detected it will print iteration count)
If no input, print error message

Valid Arguments:

-a Runs all tests
-e Runs e approximation test
-b Runs BBP pi test
-m Runs Madhava pi test
-r Runs Euler sequence pi test
-v runs Viete pi test

-n Runs Newton-Raphson sqrt test
-s Enable printing of statistics to see computed terms and iterations for each tested function.
-h : Display a help message detailing program usage.

Makefile

Bash uses shell
Set compile for C language
Setflags -Wall -Wpedantic -Werror -Wextra
Set object to be all necessary '.o' files
Generate mathlib-test executable
Compile .o files from .c files
Make mathlib-test to be main file with use of previously created object
Compile mathlib.o from mathlib.c

Credit/References/Citation:
Attended section 1/23/23
Referenced provided pseudocode from asgn2.pdf for sqrt_newton function. Used same variables