Assignment 2 – Slice of Pi

Your Name

CSE 13S – Spring 2023

Purpose

This program calculates euler's constant (*e*) and pi (π). It calculates these constants using different mathematical equations. To calculate *e*, it can use the Taylor series. To calculate π, it can use the Madhava series, the Bailey-Borwein-Plouffe formula, Viéte's formula, the Basel problem, and Wallis's formula. It can also calculate the square root of any number using the Newton-Raphson method.

How to Use the Program

To run this program, it needs to be initialized in the terminal with the corresponding command-line options. Writing '-a' after the program will run all tests. Other options include '-e' which runs the *e* approximation, '-b' which runs the Bailey-Borwein-Plouffe formula, '-m' which runs the Madhava series, '-r' which runs the Basel problem, '-v' which runs the Viéte formula, '-n' which runs the Newton-Raphson method, '-s' which prints previously computed terms, and '-h' which displays a help message. If '-s' is chosen by itself then the program will display the help message and quit.

Program Design

Data Structures

The floating-point value *epsilon* is used throughout the program to represent 10^{-14} .

For the Newton-Raphson method, the function uses a for loop that calculates each successive term of the Newton iterate.

The Taylor series function also uses a for loop to calculate each term, along with a variable that stores the new value of *k!* that is used for the denominator of each term.

The Bailey-Borwein-Plouffe formula doesn't need to use any special data structures.

The Madhava series function calls the Newton-Raphson function to find the square root of 12, and a for loop to find all the terms in the series.

The Basel problem function uses a for loop to find the summation of the series, and then calls the square root function to use on the final sum.

The Viéte function uses a for loop to multiply the product of the nested radicals. Again, the square root function from earlier is used for the radicals.

The Wallis function uses a for loop to find the factors in the multiplication series.

The file 'mathlib-test.c' uses a switch statement to find what to execute for each command line option, as well as plenty of if statements.

Algorithms

The square root function uses the equation $\frac{1}{2}$(y+$\frac{x}{y}$), with y being equal to the previous term in the series and x equal to the inputted number.

The Taylor series function uses the equation $\sum_{k=0}^{\infty} \frac{1}{k!}$.

The Bailey-Borwein-Plouffe formula uses equation $\sum_{k=0}^{\infty} \frac{1}{16^k} (\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6})$.

The Madhava series uses the equation: $\pi = \sqrt{12} \sum_{k=0}^{\infty} \frac{(-3)^{-k}}{2k+1}$.

The Basel problem uses the equation $\pi = \sqrt{6 \sum_{k=1}^{\infty} \frac{1}{k^2}}$.

The Viéte formula uses the equation $\pi = \frac{2}{\prod_{k=1}^{\infty} \frac{a_k}{2}}$.

The Wallis series uses the equation $\pi = 2 \prod_{k=1}^{\infty} \frac{4k^2}{4k^2-1}$.

Functions

The Taylor series function saves the previous k! product so that the next term only needs to multiply the previous k! by the next k number.

The Bailey-Borwein-Plouffe function uses a variable to store the product '*8k'* that is used throughout the equation. The 16 to the k power is calculated using a variable to store the previous 16^k and multiplying that by 16 to get the next term.

The Madhava series calls sqrt_newton() to calculate the rational, a variable for -3^k and then puts that in the denominator to account for the *-k*.

The Viete formula uses a variable *ak* , one for each term, and one for the product in the denominator.

The Wallis series uses a variable just to store $4k^2$.

Results

Everything works as it should in my program, and the Makefile runs without error. The only thing I could improve is the help page, but I can't think of anything else to add at the moment.