Design Document

Manikanta Illuri

Initial Design:

Objective: The object for this assignment is to implement a small number of mathematical functions (e x and p x), mimicking , and using them to compute the fundamental constants e and π.

E.c -

This file will have two functions.

The first being the e function which will use a while loop that compares a variable to the value of epsilon. Within in the loop the equation $\frac{x^k}{k!} = \frac{x^{k-1}}{(k-1)!} \times \frac{x}{k}.$ **.** I will start the variables at valuer 1 and and calculate the values based on the equation above. I will have a counter within the while loop to keep track of my iterations.

The next function will be returning the countervalue which will ve a static variable defined at the beginning of the file.

Madhava.c - I will include all the necessary files to calculate the madhava series. I will have a while loop till epsilon and within the loop I will have the value of the iterator raised to the power of 3 and then I will divide the iterator and multiply it by the value and add 1 to it.

The second function will be used to count the number of iterations the above function ran.

Euler.c - I will declare variables to keep as a counter and set that as a static variable. I will have a for loop and inside of the I will calculate the value by the square root divided by the index value iteration . Then multiply it by 10. The next function will store the value of iterations also known as the countervalue.

Bbp.c - The function will have a while loop that will run till epsilon and within the loop I will have the equation for the bbp calculation. This equation is This equation will have a for loop that will calculate summation and with in the loop I will raise the index value to the power of 16 and do the equation (k(120k +151)+47) k(k(k(512k +1024)+712)+194)+15 .

Viete.c - This will calculate the value based on the viete equation. I will have a nested for loop that will squarethe iterator value and add 2 +ak-1 to it. The nest function will calculate the iteration value to find pi.

Newton.c - I will use this function to calculate the value of pi using the Newton formula. This will have a for loop that will run based on the calculator using the 2 square root 2 divided by 9801 multiply it by 4k factorial. Then i will divide it by (k!)^4396^4k.
The second function , the second function will calculate the interactive value that we will use to calculate pi.

Mathlib-test.c -
-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.

These are the flags that I will implement in the mathlib file. These can be called when the file is being run to produce the output pi value.