Roleen Ferrer 1731261
Professor Maxwell Dunne
CSE 13S - Computer Systems and C Programming

Assignment 2: A Small Numerical Library

In this lab, I will be implementing a small numerical library which represents the sin, cosine, tangent, and exponential functions. In order to do this, I will use the pade approximations for the sin, cosine, and tangent functions and Taylor series approximations for the exponential function.

Pre-Lab Questions

Part I

See IMPLEMENTATION - Exponential Function

Part II

1. getopt() will return nothing if the values that the user inputted are valid, instead, the getopt() will go to the assigned variable using the argument. If there are invalid arguments that the user inputs or if an argument is not found, getopt() will return an error.
2. An enum would be the better choice since the assignment says that all options provided are going to be mutually exclusive.
3. See IMPLEMENTATION - Main Function

IMPLEMENTATION

Main

The main function will be primarily used to run getopt() and print the results of the function.
This function will parse through a command that is input by the user from these options:

1. -s, which runs the **sin function** test
2. -c, which runs the **cos function** test
3. -t, which runs the **tan function** test
4. -e, which runs the **exponential function** test
5. -a, which runs **all** of the tests

Pseudocode:

1) Declare an integer variable at 0 to use for the loop containing getopt()

2) Use a switch(c) case in order to parse through different options
    a) 5 cases (s, c, t, e, a) which represent the options mentioned previously
    b) Each case will go to their respective function and print an output using a for loop to parse through the given domains

Sin Function

double Sin(double pi)

In order to implement the sin function, I will be using the Pade approximation:

$$\sin(x) \approx \frac{x((x^2(52785432 - 479249x^2) - 1640635920)x^2 + 11511339840)}{((18361x^2 + 3177720)x^2 + 277920720)x^2 + 11511339840}$$

With a domain of $[-2\pi, 2\pi]$ in order to obtain better approximations.

Pseudocode:

1. Declare a variable for pi to store the value of pi from the parent function
2. Declare a variable for answer to store the answer
3. Declare a variable fo pi_squasered to stored pi * pi.
4. Use Pade approximation equation to calculate and store into the answer variable
5. Return answer

Cosine Function

double Cos(double pi)

In order to implement the cosine function, I will be using the Pade approximation:

$$\cos(x) \approx \frac{(x^2(1075032 - 14615x^2) - 18471600)x^2 + 39251520}{((127x^2 + 16632)x^2 + 1154160)x^2 + 39251520}$$

With a domain of $[-2\pi, 2\pi]$ in order to obtain better approximations.

Pseudocode:

1. Declare a variable for pi to store the value of pi from the parent function
2. Declare a variable for answer to store the answer
3. Declare a variable fo pi_squasered to stored pi * pi.
4. Use Pade approximation equation to calculate and store into the answer variable
5. Return answer

Tangent Function

In order to implement the cosine function, I will be using the Pade approximation:

$$tan(x) \approx \frac{x\,(x^2((x^2-990)\,x^2+135135)-4729725)\,x^2+34459425)}{(x^2(45x^2-13860)x^2+945945)-16216200)x^2+34459425}$$

With a domain of $[-\pi/3, \pi/3]$ in order to obtain better approximations.

Pseudocode:

double Tan(double pi)

1. Declare a variable for pi to store the value of pi from the parent function
2. Declare a variable for answer to store the answer
3. Declare a variable fo pi_squasered to stored pi * pi.
4. Use Pade approximation equation to calculate and store into the answer variable
5. Return answer

Exponential Function

In order to implement the exponential function I will be using a Taylor series approximation:

$$\frac{x^n}{n!} = \frac{x^{n-1}}{(n-1)!} \times \frac{x}{n}$$

With a domain of [0,9] in order to obtain better approximations.

Pseudocode:

double Exp(double x)

1. Declare a variable to hold the current term for the Taylor series approximation at 1
2. Declare a variable to hold the sum of the current term
3. Create a for loop to iterate through the Taylor series approximation until an accurate enough summation is achieved. The point at which this loop stops for the certain x value is determined by EPSILON, which is 10^-9
    a. for (long double k = 1.0; term > EPSILON; k += 1.0)
        i. This loop was seen in the in-class lecture on 10/20/2020
4. Return the sum of the current x iteration