Krisha Sharma

CSE13s

Winter 2022 Long

01/15/2022

Assignment 2 DESIGN.pdf

Description of Program:

This assignment is divided into 2 parts. Our first task in this assignment is to implement a small library of math functions, sin(x), cos(x), e^x, $\sqrt{x}$, and log(x). We are not allowed to use the <math.h> library to help with the first task. Each of the functions written for the first task will also halt computation using $\varepsilon = 10^{14}$. The second part of this assignment is to write a dedicated program, called integrate, that links with my implemented math library and computes numerical integrations of various functions using the composite Simpson's ⅓ rule.

Files to be included in directory "asgn2":

- functions.c
    - This file is provided and contains the implementation of the functions that my main program should integrate
- functions.h
    - This file is provided and contains the function prototypes of the functions that my main program should integrate.
- integrate.c
    - This contains the integrate() and the main() function to perform the integration specified by the command-line over the specified interval.
- mathlib.c

    - This contains the implementation of each of my math library functions.
- mathlib.h
    - This file is provided and contains the interface for my math library.
- Makefile
    - File that formats program into clang-format and compiles it into program executable “integrate” with makeintegrate/make all from Makefile.
- README.md
    - Text file in Markdown format that describes how to build and run the program, how the program handles erroneous inputs, and any problems encountered while developing the program.
- DESIGN.pdf
    - Describes design for the program thoroughly with pseudocode and visualizations.

Pseudocode / Structure:

- e^x function:
    - initialize term at 1.0

      sum = term

      k = 1

      while term > epsilon

          term *= abs(x) / k

          sum += term

          k += 1

      if x > 0, return the sum

      else 1 / x

- Notes: Initialize the term at 1.0 and set the sum equal to the term. Set k equal to 1. After defining and initializing all the variables necessary, begin the while loop. It is necessary to limit the while loop by making sure that the loop only runs if the absolute value of the term is greater than the value of epsilon (10^{-14}). Inside the loop multiply and assign the absolute value of x divided by k, to the term. Then add and assign the value of the term to the sum. Next, increase the value of k by 1. Outside the while loop, create a conditional where if x > 0, return the sum; else do 1 / x.

- sin function:
  - initialize all the variables at 1.0

    while the absolute value of the term > epsilon:

        term = term * (x * x) / ((k - 1) * k)

        sum = -sum (switch the sign)

        the current value of the series += sum * term

        k += 2.0

    return the current value of the series
  - Notes: Initialize all the necessary variables and then begin the while loop. It is necessary to limit the while loop by making sure that the loop only runs if the absolute value of the term is greater than the value of epsilon (10^{-14}). Afterwards, set the term equal to the value of (x * x) and then divide by (k - 1) * k. Change the sign of the sum, and then take the current value of the series and add and assign it to the value of sum * term. Next add and assign the value of 2.0 to k. Lastly, return the current value of the series.

- cos function:
    - The cos function will be exactly like the sin function; however, the starting point will be different. Rather than starting at 0, as in sin, the cos function will start at 1.
- $\sqrt{x}$ function:
    - initialize z at 0

      initialize y at 1

      while the absolute value of (y - z) > epsilon

      z = y

      y = 0.5 * (z + x / z)

      return y
    - Initialize the variable z at 0 and the variable y at 1 in order to account for the upper and lower limits. The z is the current term and the y is the next term. After initializing all the variables, begin coding the while loop. Limit the while loop by making sure that the loop only continues if the absolute value of (y - z) is greater than the value of epsilon. Inside the loop, set the value of z equal to the value of y, and then calculate y by doing (z + x / z) and multiplying that by 0.5 since we
- log function:
    - initialize y at 1

      p = exp(y)

      while abs(p - x) > epsilon

      y = y + x / p - 1

      p = exp(y)

      return y

    - Notes: Initialize y at 1, and set p equal to the exp(y). After initializing all the variables, begin coding the while loop and limit the loop so that the loop only runs if the absolute value of (p - x) is greater than the value of epsilon. Inside the loop set y equal to y + x / p - 1. Then set p equal to the value of exp(y). Break outside the while loop and then return y.
- integrate function
    - I have not yet gotten to the integrate function as it is in part 2 of the assignment. As soon as I have finished part 1, I will update this design document to show how I plan to do the integrate function.

Error Handling:

- I have not encountered any errors yet; however, as I program and run my code I undoubtedly will.
- This section will be updated later on.

Cite:

- I attended Eugene's section on the 14th of January, 2022 and during it he showed me how to create my Makefile as well as how to make a getopt loop when needed.