Assignment 2 - DESIGN.pdf

Description of the program:

For the DESIGN I will be focusing on the implementation of a small library of mathematical functions, as well as creating a program in C, that will use those functions in order to compute integrals. Computing integrals is the central part of the assignment, and it works by computing the numerical integration of a function over a specified interval using the composite 'Simpson's ⅓ rule'. The function will take argument inputs as well as some different command-line options, that would generate a specific function to be integrated.

Simpson's ⅓ rule:

NOTE: definition used from https://math24.net/simpsons-rule.html

Simpson's Rule is described as a numerical method that approximates the value of a definite integral with the use of quadratic functions. Simpson's Rule is based on the fact that given three points, we can find the equation of a quadratic through those points.

NOTE: image from https://www.cuemath.com/simpsons-rule-formula/

$$\Rightarrow \int_a^b f(x)dx \approx \frac{\Delta x}{3}\left[f(x_0)+4f(x_1)+2f(x_2)+\ldots..+2f(x_{n-2})+4f(x_{n-1})+f(x_n)\right]$$

where, $\Delta x = \frac{b-a}{n}$

$x_0 = a$ and $x_n = b$

$x_0, x_0, \ldots\ldots\ldots, x_n$ are the ends of the n sub intervals

$$\Rightarrow \text{Error bound} = \frac{M(b-a)^5}{180n^4}$$

where $|f^{(4)}(x)| \le M$

Files to be included:

Code Files:

1. **functions.c(provided):** This file contains the implementation of the functions for the main program

2. **functions.h:** This file contains the function prototypes of the functions integrated by the main program
3. **integrate.c:** This file contains the integrate() and the main() function to perform the integration specified by the command-line over the specified interval.
4. **mathlib.c:** This file contains the implementation of each of your math library functions.
5. **mathlib.h(provided):** This file contains the interface for my math library

Additional Files:

1. **Makefile:** formats all source code, including the header files.
2. **README.md:** Description of how to use my program and Makefile. It also includes any command-line option that my program accepts. Any false positives reported by scan-build should be documented and explained here as well. Note down any known bugs or errors in this file as well for the graders.
3. **DESIGN.pdf (This file):** Design document describes the preliminary design and design process for my program with sufficient detail for potential replication
   WRITEUP.pdf: Analysis and description of the produced program, as well as graphs generated and/or information, gathered from outputs

Pseudocode/Structure:

Mathlib.c -

In the following, we will define a math library that will help us in creating the Simpsons rule. Every expansion in this library contains 2 arguments - x and epsilon. X is the argument that will be taken into the function, and epsilon is the highest value of uncertainty (change from the actual value) that we will tolerate.

Note: Many of the following functions are taken from Assignment 2 document file

- double Exp(double x) : Returns the approximated value of e^x
  - Arguments
    - (x, epsilon = 1e-14)
  - Integers
    - trm = 1.0
    - sum = trm

        - k = 1
    - While loop
        - While trm > epsilon:
        - Trm *= abs(x) / k
        - Sum += trm
        - K += 1
    - Return statement
        - Return sum if x > 0 else 1 / sum

- double Sin(double x) Returns the approximated value of sin(x)
    - Arguments
        - (x, epsilon = 1e-14)
    - Integers
        - s = 1.0 (sign)
        - v = x (starting point)
        - t = x (records the output of the last iteration of the function)
        - K =3.0 (integer)
    - While loop
        - While abs(t) > epsilon:
        - t = t*(x * x) /((k-1)*k)
        - s = -s
        - v += s * t
        - k += 2.0
    - Return statement
        - Return v
- double Cos(double x) Returns the approximated value of cos(x)
    - Arguments
        - (x, epsilon = 1e-14)
    - Integers
        - s = 1.0 (sign)
        - v = 1.0 (Starting point)

        - t = 1.0 (records the output of the last iteration of the function)
        - k =2.0 (integer)
    - While loop
        - While abs(t) > epsilon:
        - t = t*(x * x) /((k-1)*k)
        - s = -s
        - v += s * t
        - k += 2.0
    - Return statement
        - Return v
- double Sqrt(double x) Returns the approximated value of $\sqrt{x}$
    - Arguments
        - (x, epsilon = 1e-14)
    - Integers
        - z = 0.0
        - y = 1.0
    - While loop
        - While abs(y - z) > epsilon:
        - z = y
        - y = 0.5 * (z + x / z)
    - Return statement
        - Return y
- double Log(double x) Returns the approximated value of log(x)
    - Arguments
        - (x, epsilon = 1e-14)
    - Integers
        - y = 1.0
        - p = exp(y)
    - While loop
        - While abs(p - x) > epsilon:
        - y = y + x / p - 1

        - p = exp(y)
    - Return statement
        - Return y

Implement.c -

The following is the implementation of ⅓ Simpsons rule, which is the main part of the assignment, for which we would also be using the library above:

- double integrate(function f, uint32_t l, uint32_t h, uint32_t n)
    - step = (h - l) / n
    - res = 0 <- this is where we update=-0 the result
- For loop
    - for (uint32_t i = 0; i <= n; i++)
        - x = l + (i * step)
        - fx = f(x)
        - If i == 0 or i == n(maximum value)
            - res += fx
        - Else If i % 2 != 0
            - res += 4 * fx
        - Else
            - Res += 2 * fx
- To finish we have to multiply by step/3
    - Res = (step/3) * res
- Return
    - Return res -> this should return the integral of the function inputted

The function would be able to take the following command-line options, expecting the input variable ‘x’ :

- -a: integrates $\sqrt{1 - x^4}$
- -b: integrates $\frac{1}{log(x)}$
- -c: integrates e^{-x^2}

- -d: integrates $sin(x^2)$
- -e: integrates $cos(x^2)$
- -f: integrates $log(log(x))$
- -g: integrates $\frac{sin(x)}{x}$
- -h: integrates $\frac{e^{-x}}{x}$
- -i:integrates e^{e^x}
- -j: integrates $\sqrt{sin^2(x) + cos^2(x)}$
- -n [partitions] : Sets the upper limit of partitions to use in the composite Simpson's rule to partitions, with a default value of 100
- -p [low] : Sets the low end of the interval to integrate over to low. This should not have a default value and must be specified each time the program is run.
- -q [high] : Sets the high end of the interval to integrate over to high. This should not have a default value and must be specified each time the program is run.
- -h: Displays the program's usage and synopsis. This is the 'helper' command

Credit:

- I took the majority of the pseudocode from the assignment 2 paper written by prof. Darrel Long
- I have watched the recording of Eugenes Lab section on 1/14/22, uploaded on Yuja. Unfortunately, I could not have attended the section as I am in European Time Zone. Therefore some of the structure and code were inspired from there.
- I have used external sources cited in the paper, to gain a deeper understanding of how does the Simpson ⅓ rule function (I also took the images from the sources cited)