Assignment 2 DESIGN.pdf
Yash Sharma

Description of Program:

This program computes the numerical integration of a function over a specified interval using the composite Simpson's rule. This program also contains functions that can return the approximated value of an exponential, sine, cosine, square root, and logarithm function. In other words, this is a small library of mathematical functions.

Files to be included in directory "asgn2":

1. functions.c:
    - This file contains the implementations of the math functions that the main program will use to integrate.
2. functions.h:
    - This file contains the function prototypes of the math functions that the main program will use to integrate.
3. integrate.c:
    - This file contains the integrate() and the main() function to perform the integration.
5. mathlib.c:
    - This file contains the implementation of Exponential, sine, cosine, square root, and logarithm functions. It also contains a integrate function that links the mathlib.c and computes numerical integrations of various functions using the composite Simpson's ⅓ rule.

$$\begin{aligned}\int_a^b f(x)\,dx &\approx \frac{h}{3}\sum_{j=1}^{n/2}\left[f(x_{2j-2})+4f(x_{2j-1})+f(x_{2j})\right]\\ &= \frac{h}{3}\left[f(x_0)+2\sum_{j=1}^{n/2-1}f(x_{2j})+4\sum_{j=1}^{n/2}f(x_{2j-1})+f(x_n)\right],\end{aligned}$$

6. mathlib.h:
    - This file contains the interface for the math library.

7. Makefile:
    - A file that formats program into clang-format and compiles it into program executables with make/make all from Makefile
    - Additionally, make clean from Makefile must remove compiler-generated files (such as the executables)
8. README.md:
    - This file describes how to use the program and MakeFile. It will also list and explain any command-line options the program accepts.
9. DESIGN.pdf:
    - This describes the design for the program thoroughly with pseudocode and descriptions.
10. WRITEUP.pdf:
    - This describes the program thoroughly with graphs displaying the integrated values for each number of partitions. The graphs will be generated using gnuplot and other UNIX tools. It will also include an analysis of the graphs and any lessons learned from floating-point numbers.

Pseudocode / Structure:

mathlib.c

```
Double Exp(double x) {
Set function with parameters of x and epsilon
Initialize double type variables
While the term of x is greater than epsilon, calculate the value of the term and add it to the sum
Return the value of sum if x is greater than 0 or else divide 1 by the sum
}
Double Sin(double x) {
Set function with parameters of x and epsilon
Initialize sine variables
While the absolute value of variable is greater than epsilon:
        Set variable for multiplying the first term
        Set variable to equal negative output
        Set variable to add and equal the product of two variable
        Set variable to add and equal a double
```

```
Return variable

}
Double Cos(double x) {
Set function with parameters of x and epsilon
Initialize cosine variables to 1
While the absolute value of variable is greater than epsilon:
        Set variable for multiplying the first term divided by the second term
        Set variable equal to negative variable output
        Set variable to add and equal the product of two variables
        Set variable to add and equal a double
Return variable
}

Double Sqrt(double x) {
Set function with parameters of x and epsilon
Initialize 2 variables with double
While double x is greater than 1, divide and equal 4.0 to x
Multiply and equal 2.0 to a variable
While absolute value of 2 variables subtracted is greater than epsilon
        Set 1 variable equal to the other
        Set other variable equal to formula
Return variable
}

Double Log(double x) {
Set function with parameters of x and epsilon
Initialize 2 variables, one with float and another calling the exponential function
While double x is greater than e, divide and equal e to x
Multiply and equal 1.0 to a variable
While absolute value of 2 variables subtracted is greater than epsilon
        Variable = 2 variables added divided by another variable subtracted
        Variable calls exponential function with parameter of y
Return variable y
```

```
}
```

integrate.c

```
Include header files
Create a usage function that prints out the help/manual page for using ./integrate
Double integrate(double (*f)(double), double a, double b, uint32_t n) {
    Initialize a variable to add the first part of the simpsons ⅓ rule
    Initialize another variable to add the sums of the simpsons's ⅓ rule
    Set a for loop that states the range (upper bound, lower bound) and adds the 1st sub interval
    Set another for loop that states the range and adds the 2nd sub interval
    Multiply the variable from the first part of the simpsons rule with 1 / 3
    Return the sum
Initialize a main function with arguments int and char
    Initialize opt, n, p, and q
    Set bool statements for a,b,c,d,e,f,g,h,i,j equal to false
Set a while loop that sets opt equal to get opt under the arguments and checks for the bool values
Set a switch for each statement equal to true and break
Set a default case where it displays the usage function and returns fail if function is met
Set a for loop to iterate and print out the values for each switch case
}
```

Testfile.c

```
Set a function with int main and set the argument to void
    Print out a statement that calls each function
Return 0
```

Credit:

- I attended Eugene's section on 1/14/21, which helped give me general guidance on how to approach this lab, as well as sketch out how the program runs.
- I also used the asgn2 documentation and reviewed the psuedocode provided by the professor.