Assignment 2 Design

Shirin Rokni

October 8, 2022

1 Purpose

The purpose of this program is to implement user-built trigonometric functions such as sin, cosine, arcsin, arccosine, arctangent, and log. It also compares our user-built functions to the corresponding implementation to the standard library <math.h> with a test harness. The test harness passes the functions and outputs the results into a table.

2 Mathlib-test.c

2.1 headers : needed to use some commands

#include **<stdio.h>**
#include **<unistd.h>** in order to run getopt
#include **<math.h>** to get M_PI
#include **<stdlib.h>** to get exit()
#include **"mathlib.h"** to get my functions
define the "ascSCTL" flags

2.2 sin function : prints the table for the sin function

```
define the test function for sin()
    print the x, sin, Library, Difference header
    print the lining under headers
    a for loop initializing the double float i as 0, evaluating the expression until as long as i is less than or equal to 2*pi, and updating i by incrementing it by 0.05*pi
        printing x, sin(x), the official sin(x) value, and the difference
    print a newline character for readability
    return
```

2.3 cos function : prints the table for the cos function

```
define the test function for cos()
    print the x, cos, Library, Difference header
    print the lining under headers
    a for loop initializing the double float i as 0, evaluating the expression until as long as i is less than or equal to 2*pi, and updating i by incrementing it by 0.05*pi
        printing x, cos(x), the official cos(x) value, and the difference
    print a newline character for readability
    return
```

2.4 arcsin function : prints the table for the arcsin function

```
define the test function for arcsin()
    print the x, arcsin, Library, Difference header
    print the lining under headers
    a for loop initializing the double float i as -1, evaluating the expression until as long as i is less than 1, and updating i by incrementing it by 0.05
        printing x, arcsin(x), the official arcsin(x) value, and the difference
    print a newline character for readability
    return
```

2.5 arccos function : prints the table for the arccos function

```
define the test function for arccos()
    print the x, arccos, Library, Difference header
    print the lining under headers
    a for loop initializing the double float i as -1, evaluating the expression until as long as i is less than 1, and updating i by incrementing it by 0.05
        printing x, arccos(x), the official arccos(x) value, and the difference
    print a newline character for readability
    return
```

2.6 arctan function : prints the table for the arctain function

```
define the test function for arctan()
    print the x, arctan, Library, Difference header
    print the lining under headers
    a for loop initializing the double float i as 1, evaluating the expression until as long as i is less than 10, and updating i by incrementing it by 0.05
        printing x, arctan(x), the official arctan(x) value, and the difference
    print a newline character for readability
    return
```

2.7 log function : prints the table for the log function

```
define the test function for log()
    print the x, log, Library, Difference header
    print the lining under headers
    a for loop initializing the double float i as 1, evaluating the expression until as long as i is less than 10, and updating i by incrementing it by 0.05
        printing x, log(x), the official log(x) value, and the difference
    print a newline character for readability
    return
```

2.8 main function : determines what calling certain flags would run

```
define the main function which accepts **argv as a character and argc as an integer
    defining the opt variable as 0
    a while loop which runs as long as one of the flags are called
        case 's' where the test function for sin runs
        case 'c' where the test function for cos runs
        case 'S' where the test function for arcsin runs
        case 'C' where the test function for arccos runs
        case 'l' where the test function for log runs
        case 'a' where all of the test functions are called then exits
    return 0 to prove that the program ran successfully
```

3 Mathlib.c

3.1 headers : needed to run some commands and define variables

#include **<stdio.h>**
#include **<math.h>** to get M_PI
#include **<assert.h>** to use assert
#include **"mathlib.h"** to get my functions
#define **EPSILON** as 1xE^-10

3.2 absolute value function : needed to compare values against epsilon

```
define the my_abs function accepting x as a double float
    define the variable y as an integer
    if x < y, return negative x
    else, return x
```

3.3 square root function : needed for the arctan function

```
define the my_sqrt function accepting x as a double float
    making sure that x is greater than or equal to 0
    defining the f and y variables as a doubles
    a while loop that compares x to 4
```

```
        assigning x to itself divided by 4
        assigning f to itself multiplied by 2
    defining the guess variable as a double float
    a for loop initializing the guess double float variable, evaluating if the absolute value of (y - guess) is greater than EPSILON, and updating y by dividing (y + x / y) by 2
        assigning guess to y
    return f * y
```

3.4 sin function : computes the sin of a value

```
define the my_sin function accepting x as a double float
    define the variable total, num, denom and previous as double floats
    defining the variable de_var as an integer
    assigning num/denom to previous
    a while loop that compares the absolute value of previous to EPSILON
        assigning num to itself * x * x * -1
        assigning denom to itself * de_var + 1 * de_var + 2
        assigning previous to num/denom
        assigning total to itself + previous
        incrementing de_var by 2
    return the total
```

3.4.1 cos function : computes the cos of a value

```
define the my_cos function accepting x as a double float
    define the total, num, denom and previous variable as double floats
    define the de_var variable as an integer
    assigning num/denom to previous
    a while loop that compares the absolute value of previous to EPSILON
        assigning num to itself * x * x * -1
        assigning denom to itself * (2 * de_var) * (2 * de_var - 1)
        assigning previous to num/denom
        assigning total to itself + previous
        incrementing de_var by 1
    return the total
```

3.5 arcsin function : computes the arcsin of a value

```
define the my_arcsin function accepting x as a double float
    define the previous and current variable as double floats
    define the i variable as an int
    if x is negative, make x positive and set i as negative
    a while loop that compares the absolute value of ((sin(previous) - x)/ cos(previous)) to EPSILON
```

```
        assign current to previous - ((sin(previous) - x) / cos(previous))
        assigning previous to current
    returning previous * i
```

3.6 arccos function : computes the arccos of a value

```
define the my_arccos function accepting x as a double float
    return ((pi/ 2) - sin(x))
```

3.7 arctan function : computes the arctan of a value

```
define the my_arctan function accepting x as a double float
    return arccos(1 / sqrt(x * x + 1))
```

3.8 exponential function : needed for the log function

```
define the static Exp function accepting x as a double float
    define the t and y variables as double floats
    a for loop initializing the k double float variable as 1, evaluating that t is greater than EPSILON, and updating k by incrementing it by 1
        assigning t to itself multiplied by x / k
        assigning y to itself multiplied by t
    return y
```

3.9 log function : computes the log of a value

```
define the my_log function accepting x as a double float
    define the variable excess as an integer
    define the total and e variables as double floats
    a while loop that runs if x is greater than e
        assigning x to itself divided by e
        assigning excess to itself plus an increment by 1
    define the variable diff as a double float with (x - Exp(total))/Exp(total)
    a while loop which compares the absolute value of (x - Exp(total)) with EPSILON
        assigning total to itself plus diff
        assigning diff with (x - Exp(total))/Exp(total)
    return total + excess
```