Din Bostandzic
CSE 13S

Assignment 2 - A Small Numerical Library Program Design

Pre-lab Answers

Part 1

1. Pseudocode for approximating e^x with a for loop:
   For (double k initialized to 1.0, call from math library fabs(pass in term as real) > 10e-9,
   increment k by 1.0)
   Assign to term value of (input_num/k)*term
   Increment sum by value of sum + value of term
   End for

2. Pseudocode for printing the output for e^x:
   Assuming statement is within loop that ends when total numbers (see full design below)
   Display sum with proper tabular formatting

Part 2

1. getopt() returns a variety of values depending on the situation. It will return -1 when all command-line options are parsed from the string, it will return the option character if it is found, it will return '?' if a character not defined in string is entered by the user, or it will return ':' when an option with missing argument is found.
2. Using either bool or enum can lead to similar implementations, but I have chosen bool to work better due to there being only one choice of either true or false; either the user entered the option which is in string or they did not. This can act as a status flag since each option is mutually exclusive to the others and can be checked. The enum would require checking each option as integer values, starting from 0 to the total number of options.
3. The pseudocode for the main function is provided below in the design.

The program is the implementation of the sin, cos, tan, exponential functions through the use of taylor series for each, centered at 0. The user has the ability to choose what function they would like to perform. Input values to restrict the domain of each function are used, starting from -2/pi to 2/pi inclusive (with a step of pi/16) for the sin and cos functions. As for tan the values there is restriction of [-pi/3, pi/3] with the same step as sin and cos. Finally, for exp the input values begin from 0 ending at 9 with steps of 0.1. Each function uses the taylor series which has been converted into an Pade approximation, allowing for an equation to represent the series. Each Pade approximation is put into horner normal form to allow for efficient computing and calculations. For each value the actual sin from the math library is calculated as well. This is for the use of their differences. Once the final value for each function is reached, a tabular report is displayed on screen: indicating what the function that has been called is along with all the x input values, following by columns of the sin value that has been implemented, the sin value using the math library function, and finally the difference of the two. The differences do arise because with taylor series, the more terms provided in the series, the more accurate the approximations will end up to be. However, for this program the series has ended at the 14th term which will result in these minor differences as will be discussed in the writeup.

Din Bostandzic
CSE 13S

The following functional decomposition has been implemented, along with the supporting Pseudocode:

3.0 Small Numerical Library
- 3.1 Sin()
- 3.2 Cos()
- 3.3 Tan()
- 3.4 Exp()

<u>Data Design</u>

Define EPSILON as real constant 10e-9
Define OPTIONS with string constant of “sctea”
Define SC_MIN as real constant -2*M_PI
Define SC_MAX as real constant 2*M_PI
Define SCT_STEP as real constant M_PI/16
Define TAN_MIN as real constant -M_PI/3
Define TAN_MAX as real constant M_PI/3
Define EXP_MIN as integer constant 0
Define EXP_MAX as integer constant 9
Define EXP_STEP as real constant 0.1

Declare booleans s, c, t, e, a initialized to false each

<u>Main Module Design</u>

```
Begin Main (pass in argc as integer, in argv as string)
    Declare integer c initialized to 0
    Begin While
    While (c = getopt(pass in argc as integer, in argv as string, in OPTIONS as
        string)) does not equal -1
        Begin if
        If (c == ‘s’)
            s = true
            Begin if
            If (s)
                Call Sin module()
            End if
        End if
```

Din Bostandzic
CSE 13S

```
Begin else if
Else if (c == 'c')
        c = true
        Begin if
        If (c)
                Call Cos module()
        End if
End else if
Begin else if
Else if (c == 't')
        t = true
        Begin if
        If (t)
                Call Tan module()
        End if
End else if
Begin else if
Else if (c == 'e')
        e = true
        Begin if
        If (e)
                Call Exp module()
        End if
End else if
Begin else if
Else if (c == 'a')
        a = true
        Begin if
        If (a)
                Call Sin module()
                Call Cos module()
                Call Tan module()
                Call Exp module()
        End if
End else if
Begin else if
Else if (c == '?')
        Display "Character not defined in the string"
        Return with exit status fail
End else if
```

End while
Begin If
If (argc == 1)
Display “Error: no arguments supplied!”
Return with exit status fail
End If
End Main

<u>Sin Module Data Design</u>
Declare Sin_x as real initialized to 0.0
Declare sin_lib as real initialized to 0.0

<u>Sin Module Design</u>
Begin Sin
Display “x Sin Library Difference”
Display “- --- --------- ------------”
Begin For
For (real i initialized to SC_MIN, i <= (SC_MAX+SCT_STEP), increment i by SCT_STEP)
Assign to Sin_x value of
(i*((i*i*(52785432-479249*i*i)-1640635920)*i*i+11511339840)) / (((18361*i*i+3177720)*i*i+277920720)*i*i+11511339840)
Assign to sin_lib value from call math function sin(pass in i)
Display i, Sin_x, sin_lib, (sin_lib - Sin_x)
End for
End Sin

<u>Cos Module Data Design</u>
Declare Cos_x as real initialized to 0
Declare cos_lib as real initialized to 0

<u>Cos Module Design</u>
Begin Cos
Display “x Cos Library Difference”
Display “- --- --------- ------------”
Begin For
For (real i initialized to SC_MIN, i <= (SC_MAX+SCT_STEP), increment i by SCT_STEP)

```
        Assign to Cos_x value of
        ((i*i*(1075032-14615*i*i)-18471600)*i*i+39251520) /
        (((127*i*i+16632)*i*i+1154160)*i*i+39251520)
        Assign to cos_lib value from call math function cos(pass in i)
        Display i, Cos_x, cos_lib, (cos_lib - Cos_x)
    End for
End Cos
```

Tan Module Data Design

Declare Tan_x as real initialized to 0

Declare tan_lib as real initialized to 0

Tan Module Design

```
Begin Tan
    Display "x      Tan      Library      Difference"
    Display "-      ---      ---------    ------------"
    Begin For
        For (real i initialized to TAN_MIN, i <= (TAN_MAX+SCT_STEP), increment i by
        SCT_STEP)
        Assign to Tan_x value of (i*((i*i*((i*i-990)*i*i+135135)-4729725)*i*i+34459425)) /
                    ((i*i*((45*i*i-13860)*i*i+945945)-16216200)*i*i+34459425)
        Assign to tan_lib value from call math function tan(pass in i)
        Display i, Tan_x, tan_lib, (tan_lib - Tan_x)
    End for
End Tan
```

Exp Module Data Design

Declare total_num as integer initialized to 0

Declare term as real initialized to 1.0

Declare sum as real initialized to value of term

Declare input_num as static real initialized to EXP_MIN

Declare exp_lib as real initialized to 0.0

Exp Module Design

```
Begin Exp
    Display "x      Exp      Library      Difference"
    Display "-      ---      ---------    ------------"
    Begin while
    While (total_num < 91)
        Begin For
```

```
            For (double k initialized to 1.0, call from math library fabs(pass in term as real) >
                EPSILON, increment k by 1.0)
                    Assign to term value of  (input_num/k)*term
                    Increment sum by value of sum + value of term
            End for
            Assign to exp_lib value from call math function exp(pass in input_num)
            Display input_num, sum, exp_lib, (exp_lib - sum)
            Increment input_num by EXP_STEP
            Increment total_num by 1
        End While
End Exp
```