2018

Technical Document

PROGRAMMING III

P458283

CONTENTS

Data Structure 2
Algorithms 3
Testing Procedure 5
Upgrades and Future Enhancements 5

DATA STRUCTURE

Name	Type	Purpose
storedAmount	Double	The stored amount before performing addition, subtraction, division or multiplication.
newAmount	Double	The stored amount after performing addition, subtraction, division or multiplication.
plusButtonClicked	Boolean	Let the program know that the plus button was the previously clicked button.
minusButtonClicked	Boolean	Let the program know that the minus button was the previously clicked button.
divideButtonClicked	Boolean	Let the program know that the divide button was the previously clicked button.
multiplyButtonClicked	Boolean	Let the program know that the multiply button was the previously clicked button.

ALGORITHMS

- Private void btnPlus_Click(sender, e)
    - Try
        - If either of the 4 buttons are already pressed, return nothing
        - Store the current number displayed in storedAmount
        - Clear display
        - Set the plusButtonClicked variable to true, others as false
        - Disable the 4 form buttons and enable the equals button
    - Catch
        - Display a message box
- Private void btnMinus_Click(sender, e)
    - Try
        - If either of the 4 buttons are already pressed, return nothing
        - Store the current number displayed in storedAmount
        - Clear display
        - Set the minusButtonClicked variable to true, others as false
        - Disable the 4 form buttons and enable the equals button
    - Catch
        - Display a message box
- Private void btnDivide_Click(sender, e)
    - Try
        - If either of the 4 buttons are already pressed, return nothing
        - Store the current number displayed in storedAmount
        - Clear display
        - Set the divideButtonClicked variable to true, others as false
        - Disable the 4 form buttons and enable the equals button
    - Catch
        - Display a message box
- Private void btnMultiply_Click(sender, e)
    - Try
        - If either of the 4 buttons are already pressed, return nothing
        - Store the current number displayed in storedAmount
        - Clear display
        - Set the multiplyButtonClicked variable to true, others as false
        - Disable the 4 form buttons and enable the equals button
    - Catch
        - Display a message box

- Private void btnEquals_Click(sender, e)
    - If plusButtonClicked is true, use the Math Library to add the stored amount
    - If minusButtonClicked is true, use the Math Library to minus the stored amount
    - If divideButtonClicked is true, use the Math Library to divide the stored amount
        - Prevent Infinity from happening
    - If multiplyButtonClicked is true, use the Math Library to multiply the stored amount
    - If none of the 4 are true, prevent executing further
    - Set the text display to the new amount
    - Set storedAmount and newAmount to 0
    - Set the 4 booleans to false
    - Enable the 4 form buttons and disable equals
- Private void btnSQRT_Click(sender, e)
    - Try
        - Store the current displayed number in the double “number”
        - If the number equals 0 or more
            - Use the Math Library to square root “number”
        - Display a message box and clear the text display if below 0
    - Catch
        - Display a message box for not using a double
- Private void btnCBRT_Click(sender, e)
    - Try
        - Store the current displayed number in the double “number”
        - Use the Math Library to cube root “number”
    - Catch
        - Display a message box for not using a double
- Private void btnInverse_Click(sender, e)
    - Try
        - Store the current displayed number in the double “number”
        - If the number does not equals 0
            - Use the Math Library to inverse “number”
        - Display a message box and clear the text display if equal to 0
    - Catch
        - Display a message box for not using a double
- Private void btnTan_Click(sender, e)
    - Try
        - Store the current displayed number in the double “number”
        - If the number is below 90
            - Use the Math Library to find the tangent of “number”
        - Display a message box saying the number must be below 90
    - Catch
        - Display a message box for not using a double

- Private void btnSin_Click(sender, e)
    - Try
        - Store the current displayed number in the double “number”
        - Use the Math Library to find the sine of “number”
    - Catch
        - Display a message box for not using a double
- Private void btnCos_Click(sender, e)
    - Try
        - Store the current displayed number in the double “number”
        - Use the Math Library to find the cosine of “number”
    - Catch
        - Display a message box for not using a double

TESTING PROCEDURE

There is a provided testing document that is formatted with tables to test the various functions of the program. There are 3 tables for each major functions with 3 columns for the input, expected result and real result. The test tables should use values that the program will work with correctly and values that may cause different outputs (such as using an invalid number). Evident screenshots should also be provided to show the program working.

UPGRADES AND FUTURE ENHANCEMENTS

There are a few potential upgrades/future enhancements that should be done with this program. They will be listed by priority:

1. Add more to the available calculations on the calculator, such as, to the power of 2 or power of 3.
2. Ensure calculation precision is better. Some calculations seem to not have perfect precision compared to a proper calculator.
3. Enhance the addition, subtraction, division and multiplication functions to allow for recurring calculations (e.g. 5 + 5 - 4 * 2 / 1.5).
4. Better handling of NaN or other invalid outputs.
5. Improve the GUI interface to be more compact and user friendly.