26/5/2018

Riff Ahmad Bazlee, P465225

South Metropolitan TAFE

Technical Document

Programming 3

GitHub URL: <https://github.com/RiffB/CalculatorApp>

Table of Contents

[Data Structures (CalculatorForm.cs) 1](#_Toc9931533)

[Algorithms 2](#_Toc9931534)

[Pseudocode for CalculatorApp Methods (CalculatorForm.cs) 2](#_Toc9931535)

[Constructor 2](#_Toc9931536)

[Button Click Events 2](#_Toc9931537)

[Called Functions 5](#_Toc9931538)

[CalculatorApp Error handling Techniques 11](#_Toc9931539)

[Pseudocode for Arithmetic Methods in BasicMath Library 12](#_Toc9931540)

[Arithmetic Error handling Techniques 13](#_Toc9931541)

[Pseudocode for Algebraic Methods in IntermediateMath Library 13](#_Toc9931542)

[Algebraic Error handling Techniques 13](#_Toc9931543)

[Pseudocode for Trigonometric Methods in AdvancedMath Library 14](#_Toc9931544)

[Trigonometric Error handling Techniques 14](#_Toc9931545)

[Recommended Testing 16](#_Toc9931546)

[Interface Testing 16](#_Toc9931547)

[Calculation Testing 16](#_Toc9931548)

[Recommended Upgrades 17](#_Toc9931549)

# Data Structures (CalculatorForm.cs)

|  |  |  |
| --- | --- | --- |
| Name | Type | Purpose |
| prevTotal | double | To store the previous total, to display later as part of the equation display |
| newTotal | double | To store the result of the calculation in. |
| inputValue | double | To store input, to display later as part of the equation display |
| isInputStarting | bool | To check if user input has started or not |
| isEquationStarting | bool | To check if building the equation has started or not |
| isAddLastPressed | bool | To check if the adding operator was previously pressed |
| isMinusLastPressed | bool | To check if the minus operator was previously pressed |
| isMultiplyLastPressed | bool | To check if the multiply operator was previously pressed |
| isDivideLastPressed | bool | To check if the divide operator was previously pressed |
| isSqrtLastPressed | bool | To check if the square root function was previously pressed |
| isCbrtLastPressed | bool | To check if the cube root function was previously pressed |
| isInvLastPressed | bool | To check if the inverse function was previously pressed |
| isTanLastPressed | bool | To check if the tangent function was previously pressed |
| isSinLastPressed | bool | To check if the sine function was previously pressed |
| isCosLastPressed | bool | To check if the cosine function was previously pressed |
| functionStored | string | To store function with value, to display later as part of the equation display |

# Algorithms

## Pseudocode for CalculatorApp Methods (CalculatorForm.cs)

### Constructor

|  |
| --- |
| FUNCTION CalculatorForm()  CALL InitializeComponent()  CALL ClearAndReset()  END FUNCTION |

### Button Click Events

#### Number Input Buttons

|  |
| --- |
| FUNCTION btnOne\_Click(sender, e)  CALL InputButton(btnOne.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnTwo\_Click(sender, e)  CALL InputButton(btnTwo.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnThree\_Click(sender, e)  CALL InputButton(btnThree.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnFour\_Click(sender, e)  CALL InputButton(btnFour.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnFive\_Click(sender, e)  CALL InputButton(btnFive.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnSix\_Click(sender, e)  CALL InputButton(btnSix.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnSeven\_Click(sender, e)  CALL InputButton(btnSeven.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnEight\_Click(sender, e)  CALL InputButton(btnEight.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnNine\_Click(sender, e)  CALL InputButton(btnNine.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnZero\_Click(sender, e)  CALL InputButton(btnZero.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnDecimal\_Click(sender, e)  CALL InputButton(btnDecimal.Text)  END FUNCTION |

|  |
| --- |
| FUNCTION btnNegate\_Click(sender, e)  CALL InputButton(btnNegate.Text)  END FUNCTION |

#### Clear Button

|  |
| --- |
| FUNCTION btnClear\_Click(sender, e)  CALL ClearAndReset()  END FUNCTION |

#### Function Buttons

|  |
| --- |
| FUNCTION btnSqrt\_Click(sender, e)  CALL InputFunction("sqrt")  END FUNCTION |

|  |
| --- |
| FUNCTION btnCbrt\_Click(sender, e)  CALL InputFunction("cbrt")  END FUNCTION |

|  |
| --- |
| FUNCTION btnInv\_Click(sender, e)  CALL InputFunction("inv")  END FUNCTION |

|  |
| --- |
| FUNCTION btnTan\_Click(sender, e)  CALL InputFunction("tan")  END FUNCTION |

|  |
| --- |
| FUNCTION btnSin\_Click(sender, e)  CALL InputFunction("sin")  END FUNCTION |

|  |
| --- |
| FUNCTION btnCos\_Click(sender, e)  CALL InputFunction("cos")  END FUNCTION |

#### Operative Buttons

|  |
| --- |
| FUNCTION btnAdd\_Click(sender, e)  IF CALL GetResult() returns FALSE  return  END IF  CALL UpdateDisplay(btnAdd.Text)  SET isEquationStarting to FALSE  SET isInputStarting to TRUE  CALL ResetOperatorPresses()  CALL ResetFunctionPresses()  SET isAddLastPressed to TRUE  END FUNCTION |

|  |
| --- |
| FUNCTION btnMinus\_Click(sender, e)  IF CALL GetResult() returns FALSE  return  END IF  CALL UpdateDisplay(btnMinus.Text)  SET isEquationStarting to FALSE  SET isInputStarting to TRUE  CALL ResetOperatorPresses()  CALL ResetFunctionPresses()  SET isMinusLastPressed to TRUE  END FUNCTION |

|  |
| --- |
| FUNCTION btnDivide\_Click(sender, e)  IF CALL GetResult() returns FALSE  RETURN  END IF  CALL UpdateDisplay(btnDivide.Text)  SET isEquationStarting to FALSE  SET isInputStarting to TRUE  CALL ResetOperatorPresses()  CALL ResetFunctionPresses()  SET isDivideLastPressed to TRUE  END FUNCTION |

|  |
| --- |
| FUNCTION btnMultiply\_Click(sender, e)  IF CALL GetResult() returns FALSE  RETURN  END IF  CALL UpdateDisplay(btnMultiply.Text)  SET isEquationStarting to FALSE  SET isInputStarting to TRUE  CALL ResetOperatorPresses()  CALL ResetFunctionPresses()  SET isMultiplyLastPressed to TRUE  END FUNCTION |

|  |
| --- |
| FUNCTION btnEqual\_Click(sender, e)  IF CALL GetResult() returns FALSE  RETURN  END IF  CALL UpdateDisplay(btnEqual.Text);  SET isEquationStarting to TRUE  SET isInputStarting to TRUE  CALL ResetOperatorPresses()  CALL ResetFunctionPresses()  SET prevTotal to 0.0  SET newTotal to 0.0  END FUNCTION |

### Called Functions

#### Input Functions

|  |
| --- |
| FUNCTION InputButton(buttonInput)  IF isEquationStarting is TRUE  CALL txtBxEquationDisplay.Clear()  END IF  IF CALL CheckFunctionPressed() returns TRUE  SET txtBxinputDisplay.Text to CALL FunctionDeEncapsulate(txtBxinputDisplay.Text)  END IF  IF isInputStarting is TRUE  IF txtBxinputDisplay.Text is “-0”  IF buttonInput is btnDecimal.Text  SET txtBxinputDisplay.Text to "-0" + buttonInput  ELSE IF buttonInput is btnNegate.Text  SET txtBxinputDisplay.Text to "0"  ELSE  SET txtBxinputDisplay.Text to "-" + buttonInput  END IF  ELSE  IF buttonInput is btnDecimal.Text  SET txtBxinputDisplay.Text to "0" + btnDecimal.Text  ELSE IF buttonInput is btnNegate.Text  SET txtBxinputDisplay.Text to "-0"  ELSE  SET txtBxinputDisplay.Text to buttonInput  END IF  END IF  IF buttonInput NOT btnZero.Text AND buttonInput NOT btnNegate.Text  SET isInputStarting to FALSE  END IF  ELSE  IF buttonInput is btnDecimal.Text AND txtBxinputDisplay.Text.Contains(".")  Do nothing  ELSE IF buttonInput is btnNegate.Text  DECLARE/SET double newValue …  …to Double.Parse(txtBxinputDisplay.Text) \* -1  SET txtBxinputDisplay.Text to newValue.toString()  ELSE  SET txtBxinputDisplay.Text to txtBxinputDisplay.Text + buttonInput  END IF  END IF  IF CALL CheckFunctionPressed() returns TRUE  SET txtBxinputDisplay.Text to CALL FunctionEncapsulate(txtBxinputDisplay.Text)  END IF  END FUNCTION |

|  |
| --- |
| FUNCTION InputFunction(functionInput)  IF CALL CheckFunctionPressed() returns FALSE  CALL SetFunctionBooleanTrue(functionInput)  IF isInputStarting is TRUE  SET txtBxinputDisplay.Text to "0"  END IF  SET txtBxinputDisplay.Text to CALL FunctionEncapsulate(txtBxinputDisplay.Text)  ELSE  IF CALL FunctionMatches(functionInput) returns TRUE  SET txtBxinputDisplay.Text…  …to CALL FunctionDeEncapsulate(txtBxinputDisplay.Text)  CALL ResetFunctionPresses()  ELSE  SET txtBxinputDisplay.Text…  …to CALL FunctionDeEncapsulate(txtBxinputDisplay.Text)  CALL ResetFunctionPresses()  CALL SetFunctionBooleanTrue(functionInput)  SET txtBxinputDisplay.Text…  …to CALL FunctionEncapsulate(txtBxinputDisplay.Text)  END IF  END IF  END FUNCTION |

#### Functions to assist InputFunction()

|  |
| --- |
| FUNCTION SetFunctionBooleanTrue(functionPressed)  SWITCH/CASE (functionPressed)  CASE “sqrt”  SET isSqrtLastPressed to TRUE  CASE “cbrt”  SET isCbrtLastPressed to TRUE  CASE “inv”  SET isInvLastPressed to TRUE  CASE “tan”  SET isTanLastPressed to TRUE  CASE “sin”  SET isSinLastPressed to TRUE  CASE “cos”  SET isCosLastPressed to TRUE  DEFAULT  Do nothing  END SWITCH/CASE  END FUNCTION |

|  |
| --- |
| FUNCTION bool FunctionMatches(functionPressed)  IF isSqrtLastPressed is TRUE AND functionPressed is "sqrt"  RETURN TRUE  ELSE IF isCbrtLastPressed is TRUE AND functionPressed is "cbrt"  RETURN TRUE  ELSE IF isInvLastPressed is TRUE AND functionPressed is "inv"  RETURN TRUE  ELSE IF isTanLastPressed is TRUE AND functionPressed is "tan"  RETURN TRUE  ELSE IF isSinLastPressed is TRUE AND functionPressed is "sin"  RETURN TRUE  ELSE IF isCosLastPressed is TRUE AND functionPressed is "cos"  RETURN TRUE  ELSE  RETURN FALSE  END IF  END FUNCTION |

#### Result Functions

|  |
| --- |
| FUNCTION bool GetResult()  SET prevTotal to newTotal  IF CALL CheckFunctionPressed() returns TRUE  SET functionStored to txtBxinputDisplay.Text  SET txtBxinputDisplay.Text to CALL FunctionDeEncapsulate(txtBxinputDisplay.Text)  IF CALL CalculateFunction() returns FALSE  RETURN FALSE  END IF  END IF  IF CALL CheckOperatorPressed() returns FALSE  SET newTotal to Double.Parse(txtBxinputDisplay.Text)  END IF  IF CALL CalculateOperator() returns FALSE  RETURN FALSE  END IF  RETURN TRUE  END FUNCTION |

|  |
| --- |
| FUNCTION bool CalculateOperator()  SET inputValue to Double.Parse(txtBxinputDisplay.Text)  IF isAddLastPressed is TRUE  SET newTotal to CALL BasicMath.Arithmetic.Add(prevTotal, inputValue)  RETURN TRUE  ELSE IF isMinusLastPressed is TRUE  SET newTotal to CALL BasicMath.Arithmetic.Subtract(prevTotal, inputValue)  RETURN TRUE  ELSE IF isMultiplyLastPressed is TRUE  SET newTotal to CALL BasicMath.Arithmetic.Multiply(newTotal, inputValue)  RETURN TRUE  ELSE IF isDivideLastPressed is TRUE  IF inputValue equals 0  CALL DisplayError("Cannot divide by 0")  RETURN FALSE  ELSE  SET newTotal to CALL BasicMath.Arithmetic.Divide(newTotal, inputValue)  RETURN TRUE  END IF  ELSE  RETURN TRUE  END IF  END FUNCTION |

|  |
| --- |
| FUNCTION bool CalculateFunction()  DECLARE/SET double functionValue to Double.Parse(txtBxinputDisplay.Text)  IF isSqrtLastPressed is TRUE  IF functionValue less than 0  CALL DisplayError ("Cannot SQRT Negative")  RETURN FALSE  ELSE  SET txtBxinputDisplay.Text to CALL … …IntermediateMath.Algebraic.SquareRoot(functionValue).ToString()  RETURN TRUE  END IF  ELSE IF isCbrtLastPressed is TRUE  SET txtBxinputDisplay.Text to CALL… …IntermediateMath.Algebraic.CubeRoot(functionValue).ToString()  RETURN TRUE  ELSE IF isInvLastPressed is TRUE  IF functionValue equals 0  CALL DisplayError("Cannot Divide by 0")  RETURN FALSE  ELSE  SET txtBxinputDisplay.Text to CALL… …IntermediateMath.Algebraic.Inverse(functionValue).ToString()  RETURN TRUE  END IF  ELSE IF isTanLastPressed is TRUE  IF ((functionValue / 90) + 1) % 2 equals 0  CALL DisplayError("Invalid Input")  RETURN FALSE  ELSE  SET txtBxinputDisplay.Text to CALL… …AdvancedMath.Trigonometric.Tan(functionValue).ToString()  RETURN TRUE  END IF  ELSE IF isSinLastPressed is TRUE  SET txtBxinputDisplay.Text to CALL… …AdvancedMath.Trigonometric.Sin(functionValue).ToString()  RETURN TRUE  ELSE IF isCosLastPressed is TRUE  SET txtBxinputDisplay.Text to CALL… …AdvancedMath.Trigonometric.Cos(functionValue).ToString()  RETURN TRUE  ELSE  RETURN TRUE  END IF  END FUNCTION |

#### Display Functions

|  |
| --- |
| FUNCTION UpdateDisplay(operativePressed)  IF isEquationStarting is TRUE  IF CALL CheckFunctionPressed() returns TRUE  SET txtBxEquationDisplay.Text to…  …functionStored + operativePressed  ELSE  SET txtBxEquationDisplay.Text to…  … inputValue.ToString() + operativePressed  END IF  ELSE  IF isAddLastPressed is TRUE  SET txtBxEquationDisplay.Text to…  … "(" + prevTotal.ToString() + ")" + btnAdd.Text +…  …inputValue.ToString() + btnEqual.Text  ELSE IF isMinusLastPressed is TRUE  SET txtBxEquationDisplay.Text to…  … "(" + prevTotal.ToString() + ")" + btnMinus.Text +…  …inputValue.ToString() + btnEqual.Text  ELSE IF isMultiplyLastPressed is TRUE  SET txtBxEquationDisplay.Text to…  … "(" + prevTotal.ToString() + ")" + btnMultiply.Text +…  …inputValue.ToString() + btnEqual.Text  ELSE IF isDivideLastPressed is TRUE  SET txtBxEquationDisplay.Text to…  … "(" + prevTotal.ToString() + ")" + btnDivide.Text +…  …inputValue.ToString() + btnEqual.Text  END IF  END IF  SET txtBxinputDisplay.Text to newTotal.ToString()  END FUNCTION |

|  |
| --- |
| FUNCTION string FunctionEncapsulate(value)  IF isSqrtLastPressed is TRUE  SET value to "SQRT(" + value + ")"  ELSE IF isCbrtLastPressed is TRUE  SET value to "CBRT(" + value + ")"  ELSE IF isInvLastPressed is TRUE  SET value to "INV(" + value + ")"  ELSE IF isTanLastPressed is TRUE  SET value to "TAN(" + value + ")"  ELSE IF isSinLastPressed is TRUE  SET value to "SIN(" + value + ")"  ELSE IF isCosLastPressed is TRUE  SET value to "COSE(" + value + ")"  END IF  RETURN value  END FUNCTION |

|  |
| --- |
| FUNCTION string FunctionDeEncapsulate(value)  IF isSqrtLastPressed is TRUE OR isCbrtLastPressed is TRUE  SET value to value.Substring(5, value.Length - 6)  ELSE  SET value to txtBxinputDisplay.Text.Substring(4, value.Length - 5)  END IF  RETURN value  END FUNCTION |

#### Boolean Functions

|  |
| --- |
| FUNCTION bool CheckFunctionPressed()  IF isSqrtLastPressed is TRUE OR isCbrtLastPressed is TRUE OR…  … isInvLastPressed is TRUE OR isTanLastPressed is TRUE OR…  … isSinLastPressed is TRUE OR isCosLastPressed is TRUE  RETURN TRUE  ELSE  RETURN FALSE  END IF  END FUNCTION |

|  |
| --- |
| FUNCTION bool CheckOperatorPressed()  IF isAddLastPressed is TRUE OR isMinusLastPressed is TRUE OR…  … isMultiplyLastPressed is TRUE OR isDivideLastPressed is TRUE  RETURN TRUE  ELSE  RETURN FALSE  END IF  END FUNCTION |

|  |
| --- |
| FUNCTION ResetOperatorPresses()  SET isAddLastPressed to FALSE  SET isMinusLastPressed to FALSE  SET isMultiplyLastPressed to FALSE  SET isDivideLastPressed to FALSE  END FUNCTION |

|  |
| --- |
| FUNCTION ResetFunctionPresses()  SET isSqrtLastPressed to FALSE  SET isCbrtLastPressed to FALSE  SET isInvLastPressed to FALSE  SET isTanLastPressed to FALSE  SET isSinLastPressed to FALSE  SET isCosLastPressed to FALSE  END FUNCTION |

#### Error message and Clearing Functions

|  |
| --- |
| FUNCTION DisplayError(message)  SET txtBxinputDisplay.Text to message  SET prevTotal to 0.0  SET newTotal to 0.0  SET inputValue to 0.0  SET isEquationStarting to TRUE  SET isInputStarting to TRUE  CALL ResetOperatorPresses()  CALL ResetFunctionPresses()  SET functionStored to “”  END FUNCTION |

|  |
| --- |
| FUNCTION ClearAndReset()  SET txtBxinputDisplay.Text to btnZero.Text  CALL txtBxEquationDisplay.Clear()  SET prevTotal to 0.0  SET newTotal to 0.0  SET inputValue to 0.0  SET isEquationStarting to TRUE  SET isInputStarting to TRUE  CALL ResetOperatorPresses()  CALL ResetFunctionPresses()  SET functionStored to “”  END FUNCTION |

## CalculatorApp Error handling Techniques

In the CalculateOperator() function, under the ELSE IF isDivideLastPressed is TRUE block. The following was used to catch if it was dividing by 0.

|  |
| --- |
| IF inputValue equals 0  CALL DisplayError("Cannot divide by 0")  RETURN FALSE  ELSE  SET newTotal to CALL BasicMath.Arithmetic.Divide(newTotal, inputValue)  RETURN TRUE  END IF |

In the CalculateFunction() function, the following was used to catch these errors: Square root a negative number, Inversing a 0, Finding tan value of 90, 270, -90 etc.

In the IF isSqrtLastPressed is TRUE block:

|  |
| --- |
| IF functionValue less than 0  CALL DisplayError ("Cannot SQRT Negative")  RETURN FALSE  ELSE  SET txtBxinputDisplay.Text…  …to CALL IntermediateMath.Algebraic.SquareRoot(functionValue).ToString()  RETURN TRUE  END IF |

In the ELSE IF isInvLastPressed is TRUE block

|  |
| --- |
| IF functionValue equals 0  CALL DisplayError("Cannot Divide by 0")  RETURN FALSE  ELSE  SET txtBxinputDisplay.Text to CALL… …IntermediateMath.Algebraic.Inverse(functionValue).ToString()  RETURN TRUE  END IF |

In the ELSE IF isTanLastPressed is TRUE block

|  |
| --- |
| IF ((functionValue / 90) + 1) % 2 equals 0  CALL DisplayError("Invalid Input")  RETURN FALSE  ELSE  SET txtBxinputDisplay.Text to CALL… …AdvancedMath.Trigonometric.Tan(functionValue).ToString()  RETURN TRUE  END IF |

## Pseudocode for Arithmetic Methods in BasicMath Library

|  |
| --- |
| FUNCTION double Add(a,b)  RETURN a+b  END FUNCTION |

|  |
| --- |
| FUNCTION double Subtract(a,b)  RETURN a-b  END FUNCTION |

|  |
| --- |
| FUNCTION double Multiply(a,b)  RETURN a\*b  END FUNCTION |

|  |
| --- |
| FUNCTION double Divide(a,b)  IF b equals 0.0  THROW new Exception()  ELSE  RETURN a/b  END IF  END FUNCTION |

## Arithmetic Error handling Techniques

In the Divide(a,b) Method, Error was handle if divisor(b) equals 0

|  |
| --- |
| IF b equals 0.0  THROW new Exception()  ELSE  RETURN a/b  END IF |

## Pseudocode for Algebraic Methods in IntermediateMath Library

|  |
| --- |
| FUNCTION double SquareRoot(a)  IF a less than 0.0  THROW new Exception()  ELSE  RETURN Math.Pow(a, (double)1 / 2)  END IF  END FUNCTION |

|  |
| --- |
| FUNCTION double CubeRoot(a)  RETURN Math.Pow(Math.Abs(a), (double)1 / 3) \* (Math.Sign(a))  END FUNCTION |

|  |
| --- |
| FUNCTION double Inverse(a)  IF a equals 0.0  THROW new Exception()  ELSE  RETURN 1 / a  END IF  END FUNCTION |

## Algebraic Error handling Techniques

In the SquareRoot(a) Method, an error was handled if the value is a negative number

|  |
| --- |
| IF a less than 0.0  THROW new Exception()  ELSE  RETURN Math.Pow(a, (double)1 / 2)  END IF |

In the Inverse(a) Method, an error was handled if the value was 0

|  |
| --- |
| IF a equals 0.0  THROW new Exception()  ELSE  RETURN 1 / a  END IF |

## Pseudocode for Trigonometric Methods in AdvancedMath Library

|  |
| --- |
| FUNCTION double Tan(a)  IF ((a / 90) + 1) % 2 equals 0.0  THROW new Exception()  ELSE  IF a % 180 equals 0.0  RETURN 0.0  ELSE  RETURN Math.Tan(a \* Math.PI / 180)  END IF  END IF  END FUNCTION |

|  |
| --- |
| FUNCTION double Sin(a)  IF a % 180 equals 0.0  RETURN 0.0  ELSE  RETURN Math.Sin(a \* Math.PI / 180)  END IF  END FUNCTION |

|  |
| --- |
| FUNCTION double Cos(a)  IF ((a / 90) + 1) % 2 equals 0.0  RETURN 0.0  ELSE  RETURN Math.Cos(a \* Math.PI / 180)  END IF  END FUNCTION |

## Trigonometric Error handling Techniques

In the Tan(a) Method, an error was handled if finding the Tangent of 90, -90, 270 etc degrees

|  |
| --- |
| IF ((a / 90) + 1) % 2 equals 0.0  THROW new Exception()  ELSE  IF a % 180 equals 0.0  RETURN 0.0  ELSE  RETURN Math.Tan(a \* Math.PI / 180)  END IF  END IF |

Also in all of the Trigonometric Method, there is a bug where it will not return 0 for the correct degrees inputted. This error was handled to return 0.0 IF the correct entered degrees was entered.

In Tan(a):

|  |
| --- |
| IF a % 180 equals 0.0  RETURN 0.0  ELSE  RETURN Math.Tan(a \* Math.PI / 180)  END IF |

In Sin(a):

|  |
| --- |
| IF a % 180 equals 0.0  RETURN 0.0  ELSE  RETURN Math.Sin(a \* Math.PI / 180)  END IF |

In Cos(a)

|  |
| --- |
| IF ((a / 90) + 1) % 2 equals 0.0  RETURN 0.0  ELSE  RETURN Math.Cos(a \* Math.PI / 180)  END IF |

# Recommended Testing

The recommended Testing Procedure for this program are divided into testing the interface, and testing the calculations.

## Interface Testing

Interface Testing would go into testing each button on the form to see that it works as intende. It is recommended for testing to go further to test button combinations that will often pressed in sequence.

## Calculation Testing

Because the Calculator only calculates two values together, i.e, just continues on from the previous result, there is no testing for BIMDAS/order of operations. Testing should be made for each operations and functions as follows:

|  |  |
| --- | --- |
| TestID | Test Description |
| ADD1 | Adding two negative numbers |
| ADD2 | Adding numbers to get a negative result |
| ADD3 | Adding a positive and negative number together |
| ADD4 | Adding two positive numbers together |
| SUB1 | Subtracting two negative numbers together |
| SUB2 | Subtracting two numbers to get a positive result |
| SUB3 | Subtracting two numbers to get a negative result |
| SUB4 | Subtracting a positive and negative number together |
| MUL1 | Multiplying two negative numbers together |
| MUL2 | Multiplying two numbers to get a positive result |
| MUL3 | Multiplying a positive and negative number together |
| DIV1 | Divide a large number with a smaller number |
| DIV2 | Divide a small number with a larger number |
| DIV3 | Divide to get a recurring decimal result |
| DIV4 | Divide with 2 negative numbers |
| DIV5 | Divide with a positive and a negative number together |
| DIV6 | Error Catch: Divide by zero |
| SQR1 | Square root 0 |
| SQR2 | Square root a positive number |
| SQR3 | Error Catch: Square root a negative number |
| CBR1 | Cube root 0 |
| CBR2 | Cube root a positive number |
| CBR3 | Cube root a negative number |
| INV1 | Error Catch: Inverse of 0 |
| INV2 | Inverse of a positive number |
| INV3 | Inverse of a negative number |
| TAN0 | Tangent of 0 degrees |
| TAN30 | Tangent of 30 degrees |
| TAN60 | Tangent of 45 degrees |
| TAN90 | Error Catch: Tangent 90 degrees |
| TAN180 | Tangent if 180 degrees |
| TAN270 | Error Catch: Tangent of 270 degrees |
| SIN0 | Sine of 0 degrees |
| SIN30 | Sine of 30 degrees |
| SIN60 | Sine of 60 degrees |
| SIN90 | Sine of 90 degrees |
| SIN180 | Sine of 180 degrees |
| SIN270 | Sine of 270 degrees |
| COS0 | Cosine of 0 degrees |
| COS30 | Cosine of 30 degrees |
| COS60 | Cosine of 60 degrees |
| COS90 | Cosine of 90 degrees |
| COS180 | Cosine of 180 degrees |
| COS270 | Cosine of 270 degrees |

# Recommended Upgrades

Improvements can be made for this calculator such as:

* Adding keyboard/keypad input
* Adding more values for the equation to hold instead of just 2
* Adding BIMDAS order of operations along with bracket input buttons
* When clicking operation buttons in sequence, it will not reset the intended calculation first, but instead calculate using the first entered operation, then input the second as the next intended operation. This may be fixed by adding another if block to check if input has started when clicking on operation buttons. E.G If input hasn’t started, then don’t calculate yet.