W

Programming 3 AT-1.7

Technical Documentation

chris kalms

P465754@tafe.wa.edu.au

GitHub: <https://github.com/ChrisKalms/Calculator/>

Contents

[Data Structure 2](#_Toc10578525)

[Algorithms 3](#_Toc10578526)

[Trigonometry 3](#_Toc10578527)

[DegreesTan(double degrees, out double tan) : bool 3](#_Toc10578528)

[DegreesSin(double degrees) : double 3](#_Toc10578529)

[DegreesCos(double degrees) : double 4](#_Toc10578530)

[ConvertToRadians(double degrees) : double 4](#_Toc10578531)

[Algebra 5](#_Toc10578532)

[SquareRoot(double valueToSquareRoot, out double output) : bool 5](#_Toc10578533)

[CubeRoot(double valueToCubeRoot) : double 5](#_Toc10578534)

[Inverse(double valueToBeInverted, out double output) : bool 5](#_Toc10578535)

[Arithmetic 6](#_Toc10578536)

[Addition(double addTo, double additionAmount) : double 6](#_Toc10578537)

[Subtraction(double subtractFrom, double subtractAmount) : double 6](#_Toc10578538)

[Division(double divideFrom, double divisionAmount, out double result) : bool 6](#_Toc10578539)

[Multiplication(double toBeMultiplied, double multiplyAmount) : double 6](#_Toc10578540)

[Recommended Upgrades/Enhancements 7](#_Toc10578541)

# Data Structure

|  |  |  |
| --- | --- | --- |
| Name: | Type: | Purpose: |
| valueHolder | Double | Holds the output value from equations |
| total | Double | Holds the output value after = is clicked |
| isTotal | Bool | Used to indicate how to treat values being inputted. |
| plusButtonClicked | Bool | Let = know that + was the last button pressed. |
| minusButtonClicked | Bool | Let = know that - was the last button pressed. |
| divisionButtonClicked | Bool | Let = know that / was the last button pressed. |
| multiplyButtonClicked | Bool | Let = know that \* was the last button pressed. |

# Algorithms

## Trigonometry

### DegreesTan(double degrees, out double tan) : bool

If degrees % 90 == 0 && degrees/90 % 2 != 0

Return false

Else if degrees % 90 == 0

tan = 0

else

Tan = mathtan(degrees as radians)

Return true

Degrees Tan Returns Boolean so that the method can sanitize data. Due to some inputs such as 90degrees tan being undefined. This means that the method can return false whenever an undefined return is imminent. The algorithm checks to see if the input is multiples of 90 and sets the return value there. If the input isn’t a multiple of 90 .net Math.Tan() is called on the input and the output is returned.

### DegreesSin(double degrees) : double

If degrees == 0

return 0

Else if degrees % 90 == 0 && degrees/90 % 2 == 0

return 0

Else if degrees % 90 == 0 && (degrees/90 – 1) % 4 == 0)

return 1

Else if degrees % 90 == 0

return -1

else

Return mathsin(degrees as radians)

DegreesSin returns a double, it doesn’t need to have a return type of bool as there are no occurrences of undefined in Sine. The algorithm needs to check the input for being divisible by 90 so that it can ensure correct output. If the input isn’t divisible by 90 than .net Math.Sin() is called and the output is returned.

### DegreesCos(double degrees) : double

If degrees == 0

Return 1

If degrees % 90 == 0 && degrees/90 % 2 == 0

If degrees / 180 % 2 == 0

Return 1

else

Return -1

else if degrees % 90 == 0

return 0

else

Return mathcos(degrees as radians)

DegreesCos returns a double, it doesn’t need to have a return type of bool as there are no occurrences of undefined in Cosine. The algorithm needs to check the input for being divisible by 90 so that it can ensure correct output. If the input isn’t divisible by 90 than .net Math.Cos() is called and the output is returned.

### ConvertToRadians(double degrees) : double

Return degrees \* PI / 180

ConvertToRadians is a helper method that returns a double. It takes in an amount in degrees and converts it into radians, this value is then returned.

## Algebra

### SquareRoot(double valueToSquareRoot, out double output) : bool

output = √valueToSquareRoot

if output isnan || infinity

return false

else

return true

SquareRoot takes an input and tries to perform the square root function on it. Due to instances of SquareRoot returning undefined such as √-1 , where this is the case, the method can return false. If the square root function was successful, the method will return true and output the result through the out double output.

### CubeRoot(double valueToCubeRoot) : double

If valueToCubeRoot < 0

Perform cuberoot as a positive and convert back to a negative

Return output

else

Return cuberoot

CubeRoot returns a double as there are no occurrences where the output can be undefined. The method takes an input and performs the cube root function on it. In the instance of a negative number being input, the algorithm needs to convert it into a positive number before performing the cube root function, after which the value needs to be returned to a negative number. The value is then output.

### Inverse(double valueToBeInverted, out double output) : bool

output = 1 / valueToBeInverted

if output isnan || infinity

return false

else

return true

Inverse returns a Boolean value as exception handling needs to take place where zero is input. This effectively turns into a divide by zero exception and can return false and not interrupt the flow of the program calling it. If the input is valid, the return method will return true and return the output through the out double output.

## Arithmetic

### Addition(double addTo, double additionAmount) : double

Return addTo + additionAmount

Addition returns a double. The method takes in two values and adds them together, the result is then returned.

### Subtraction(double subtractFrom, double subtractAmount) : double

Return subtractFrom – subtractAmount

Subtraction returns a double. The method takes in two values and subtracts the second value from the first. The result is then returned.

### Division(double divideFrom, double divisionAmount, out double result) : bool

if divisionAmount == 0

return false

else

result = divideFrom / divisionAmount

Division returns a Boolean. The method takes in two inputs and divides the first input by the second input. The algorithm checks to see if the second input is a zero and returns false to the caller. Due to divide by zero being undefined this is necessary. If the second input isn’t a zero, the division occurs as normal, the output is returned through out double result and true is returned from the method.

### Multiplication(double toBeMultiplied, double multiplyAmount) : double

Return toBeMultiplied \* multiplyAmount

Multiplication returns a double. The method takes in two inputs the first being the amount to be multiplied, the second the amount to multiply the first input by. The output is then returned.

# Recommended Testing Procedure

This software should be tested to ensure that calculations, especially where ‘undefined’ is a possibility are acting as specified.

Calculations where ‘undefined’ is a possibility are:

* DegreesTan(90)
* Division(0)
* SquareRoot(-1)
* Inverse(0)

Recommended Test Table

|  |  |  |
| --- | --- | --- |
| Test Name. | Expected Result | Outcome |
| Addition (25 + 4) | 29 |  |
| Subtraction (29 – 4) | 25 |  |
| Multiplication (30 \* 3) | 90 |  |
| Division (90 / 3) | 30 |  |
| Division (25/0) | Error |  |
| Square Root 9 | 3 |  |
| Square Root -1 | Error |  |
| Cube Root 27 | 3 |  |
| Cube Root -8 | -2 |  |
| Inverse 10 | .1 |  |
| Inverse 0 | Error |  |
| Sin 25º | 0.422618 … |  |
| Sin 90º | 1 |  |
| Cos 45º | 0.707106 … |  |
| Cos 90º | 0 |  |
| Tan 45º | 1 |  |
| Tan 90º | Error |  |
| Tan 180º | 0 |  |

# Recommended Upgrades/Enhancements

1. Add Functionality After isTotal is set.

* Currently after an event triggering isTotal it essentially resets the calculator. This means that you can’t perform calculations on the result of a calculation.

1. Add +/- button

* The minus button currently sets a negative number in the text box
* The functionality of this is limited due to once a number is in the field it’s merely stored in the valueHolder variable and ready for subtraction.

1. Reorganize the if else statements in the Trigonometry class so that a faster result is returned if the input is not divisible by 90.

* Currently it runs through all checks before performing operation on input that’s not divisible by 90. There should be: if(divisible by 90){…} else return Math.(sin, cos, tan).