DEPARTMENT OF INFORMATION TECHNOLOGY CENTER FOR DIPLOMA STUDIES PROJECT REPORT DAT 21003: VISUAL PROGRAMMING

PROJECT REPORT DAT 21003: VISUAL PROGRAMMING

TITLE	SCIENTIFIC CALCULATOR
COURSE	DAT / VISUAL PROGRAMMING
STUDENT NAME	MOHAMAD HAIKAL EKMAL BIN YUSOF (AA212066) MUHAMAD HAZRIN HAKIM BIN HAZINOL (AA211500) MUHAMMAD SHUKRI BIN AMAN (AA211426) MUHAMMAD HAZIQ RAIMI BIN MD KHAIRUL (AA21147) MUHAMMAD A'KIF BIN MOHD AZMI (AA210003)
LECTURER NAME	TS. DR. NOORDIANA BINTI KASSIM@KASIM
SECTION / GROUP	8 / 5

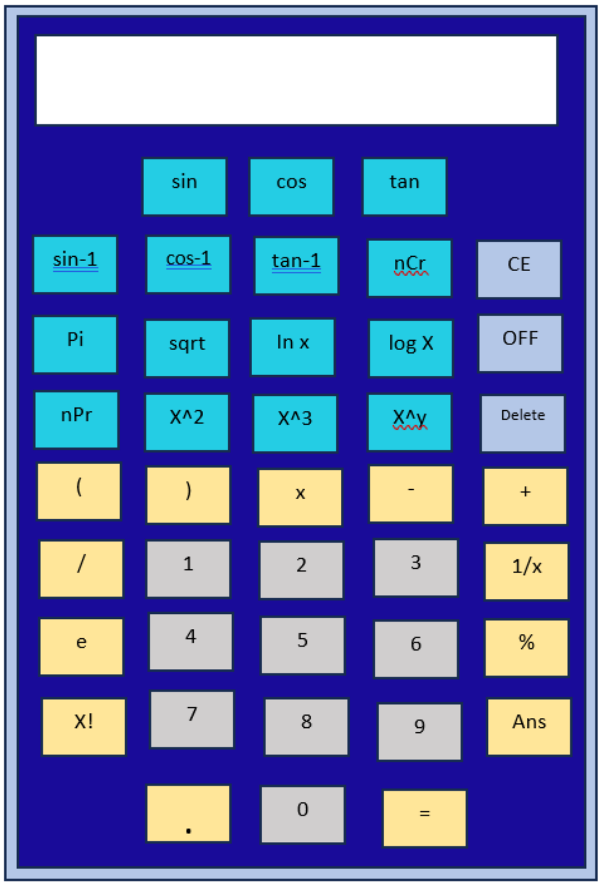
TABLE OF CONTENTS

ANALYSIS

DESIGN DESCRIPTION

a) Interface

Figure 1: Sketch of scientific calculator interface



b) Algorithm (pseudocode)

i. Procedures

1) <u>Event</u> -Creating following event in My Calculator:

1. Event Click for *btnSin*:
   i. Display…….at txtDisplay.Text
2. Event Click for *btnCos*:
   i. Display…….at txtDisplay.Text
3. Event Click for *btnTan*:
   i. Display…….at txtDisplay.Text
4. Event Click for *btnSin1*:
   i. Display…….at txtDisplay.Text
5. Event Click for *btnCos1*:
   i. Display…….at txtDisplay.Text
6. Event Click for *btnTan1*:
   i. Display…….at txtDisplay.Text
7. Event Click for *btnX*:
   i. Display…….at txtDisplay.Text
8. Event Click for *btnSqrt*:
   i. Display…….at txtDisplay.Text
9. Event Click for *btnLog*:
   i. Display…….at txtDisplay.Text
10. Event Click for *btnInx:*
    i. Display…….at txtDisplay.Text
11. Event Click for *btneE*:
    i. Display…….at txtDisplay.Text

12. Event Click for *btnnPr*:
    i. Display…….at txtDisplay.Text

13. Event Click for *btnPi*:
    i. Display…….at txtDisplay.Text

14. Event Click for *btnBracket2*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at button40.Text

15. Event Click for *btnBracket1*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at button40.Text;

16. Event Click for *btnBackspace*:
    i. Display…….at txtDisplay.Text

17. Event Click for *btnnCr*:
    i. Display…….at txtDisplay.Text

18. Event Click for *btnDevide*:
    i. Display…….at txtDisplay.Text

19. Event Click for *btnMultiply*:
    i. Display…….at txtDisplay.Text

20. Event Click for *btnMinus*:
    i. Display…….at txtDisplay.Text

21. Event Click for *btnAdd*:
    i. Display…….at txtDisplay.Text

22. Event Click for *btnX2*:
    i. Display…….at txtDisplay.Text

23. Event Click for *btnX3*:
    i. Display…….at txtDisplay.Text

24. Event Click for *btnXy*:
    i. Display…….at txtDisplay.Text

25. Event Click for *btn1x*:
    i. Display…….at txtDisplay.Text

26. Event Click for *btnPercent*:
    i. Display…….at txtDisplay.Text

27. Event Click for *btnAns*:
    i. Display…….at txtDisplay.Text

28. Event Click for *btnAnswer*:
    i. Display…….at txtDisplay.Text

29. Event Click for *btnPoint*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnPoint.Text

30. Event Click for *btnNum7*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnNum7.Text

31. Event Click for *btnNum8*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnNum8.Text

32. Event Click for *btnNum9*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnNum9.Text

33. Event Click for *btnNum4*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnNum4.Text

34. Event Click for *btnNum5*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnNum5.Text

35. Event Click for *btnNum6*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnNum6.Text

36. Event Click for *btnNum1*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnNum1.Text

37. Event Click for *btnNum2*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnNum2.Text

38. Event Click for *btnNum3*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnNum3.Text

39. Event Click for *btnNum0*:
    i. Display…….at txtDisplay.Text
    ii.Display…….at btnNum0.Text

40. Event Click for *btnOFF*:
    i. Display…….at Exit

41. Event Click for *btnCE*:
    i. Display…….at txtDisplay.Text

2) Function- Have 6 functions in My Calculator code

1. Functions:

‘Calculate’: Calculates the result of a mathematical expression given as a string.
Syntax = static public double Calculate(string input)

2. Private Functions:

‘GetExpression’: Processes the input string to extract the mathematical expression.
Syntax = static private string GetExpression(string input)

‘Counting’: Evaluates the processed expression to obtain the final result.
Syntax = static private double Counting(string input)

3. Helper Functions:

‘IsDelimeter’: Checks if a character is a delimiter.
Syntax = static private bool IsDelimeter(char c)

‘IsOperator’: Checks if a character is an operator.
Syntax = static private bool IsOperator(char c)

‘IsFunction’: Checks if a string represents a valid function.
Syntax = static private bool IsFunction(String s)

‘doFunc’: Performs a mathematical function on a given parameter.
Syntax = static private String doFunc(String fun, double param)

‘GetPriority’: Assigns priorities to operators.
Syntax = static private byte GetPriority(char s)

‘factorial’: Calculates the factorial of a number.
Syntax = private static int factorial(int x)

ii. Control Structures

a) <u>Selection</u>

1. if statement in the Calculate method:
   Explanation: This if statement checks if the input string can be parsed directly into a double. If it can, the method returns the parsed double value. Otherwise, it proceeds to the next step for further processing.

2. if-else statement in the Calculate method:
   Explanation: This if-else statement catches an exception when parsing the input string into a double and then calls the Counting method to perform further calculations.

3. if-else statement in the GetExpression method:
   Explanation: This if-else statement checks the input characters to identify numbers, operators, and negative numbers in the input string.

4. if-else statement in the Counting method:
   Explanation: This if-else statement handles different cases for operators and performs corresponding mathematical operations.

5. if-else statement in the IsDelimeter method:
   Explanation: This if-else statement checks if a character is a delimiter (space or equals sign).

6. if statement in the IsOperator method:
   Explanation: This if statement checks if a character is an operator (+, -, *, /, ^, (,), P, C, !, or %).

7. if statement in the IsFunction method:
   Explanation: This if statement checks if a string corresponds to a valid mathematical function.

8. switch statement in the Counting method:
Explanation: This switch statement is used to handle different cases for operators (+, -, *, /, ^, !, %) and perform the corresponding mathematical operations.

b) Iteration

1. Iteration 1: for loop in the GetExpression method
= **Purpose**: This loop iterates over each character of the modified input string (input) to tokenize the expression by separating operators, functions, and operands.
= **Explanation**: Similar to the previous loop, this loop starts at index 0 and continues until the end of the input string. It analyzes each character and constructs the output string by adding the appropriate tokens.

2. Iteration 2: while loop in the Counting method
= **Purpose**: This loop iterates over each character of the input string to perform calculations and evaluate the expression.
= **Explanation**: The loop continues as long as there are characters remaining in the input string. It processes each character and performs the corresponding mathematical operations, such as pushing operands to a stack and applying operators.

3. Iteration 3: for loop in the Counting method
= **Purpose**: This loop iterates over each character of the input string to handle specific cases and calculate the final result.
= **Explanation**: The loop starts at index 0 and continues until the end of the input string. It analyzes each character and performs calculations based on the type of character, such as pushing operands to a stack, applying operators, or handling special mathematical functions

c) Array

1.String array func in the IsFunction method:
Explanation: This array holds a list of valid mathematical functions such as "sin", "cos", "tg", "asin", "acos", "atg", "sqrt", "ln", "lg". It is used to check if a given string represents a valid function.

iii. Exception Handling

```
try { return double.Parse(GetExpression(input)); }
catch (Exception) { return Counting(GetExpression(input)); }
```

The code tries to convert the input string into a number using double.Parse(). This is done in the try block. If the conversion is successful, it returns the parsed number.

However, if an error occurs during the conversion (for example, if the input is not a valid number), an exception is thrown. The catch block is used to handle this exception. Instead of giving up, the code calls the GetExpression(input) method again to get an alternative expression. This alternative expression is then passed to the Counting() method, which performs a different type of calculation.

Development

a. User interface

The user will be directed to the scientific calculator's page only and the flow of operations will run at the same page including the outputs based on **Figure 3**.

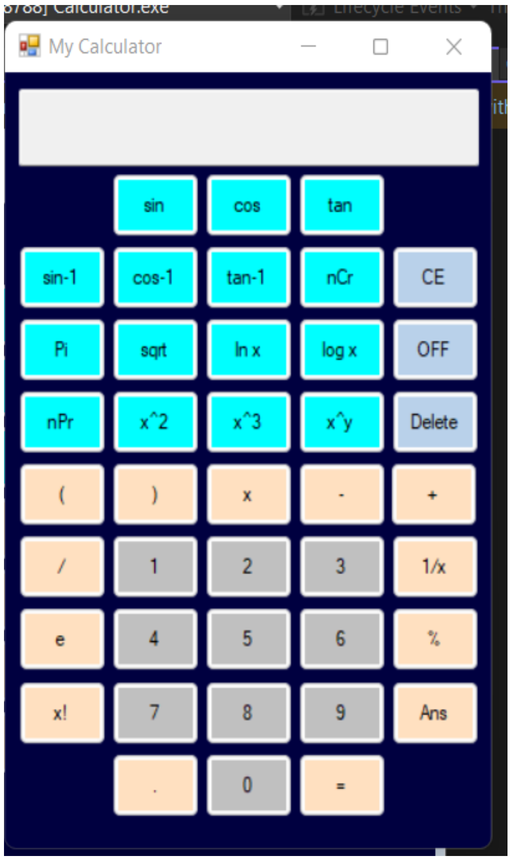


Figure 3: User interface

TEST

1. Sample Input & Output

The user inserts numbers first by clicking on the number then it will display on the textbox at the top. Once the first number has been inserted, the users need to click on any operation symbols of the calculator. Then, users need to select the second number and click the equation or answer (=/Ans) button at the bottom to run the calculated process based on Figure 5. If the inputs and calculation are correct, the output will display on the same textbox based on Figure 6.

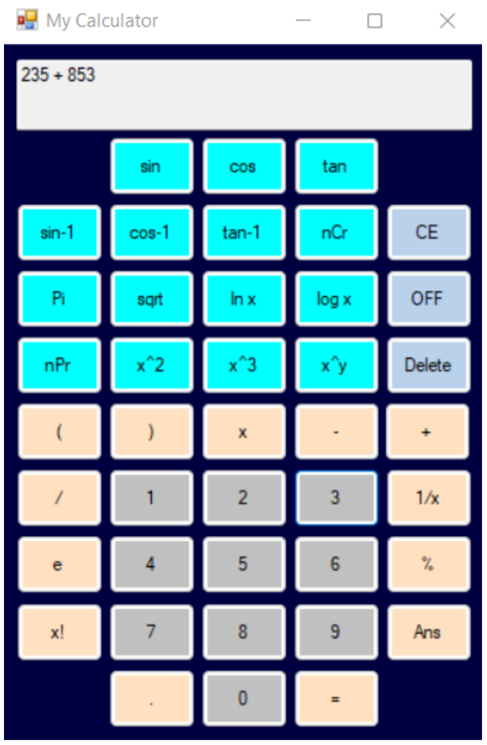
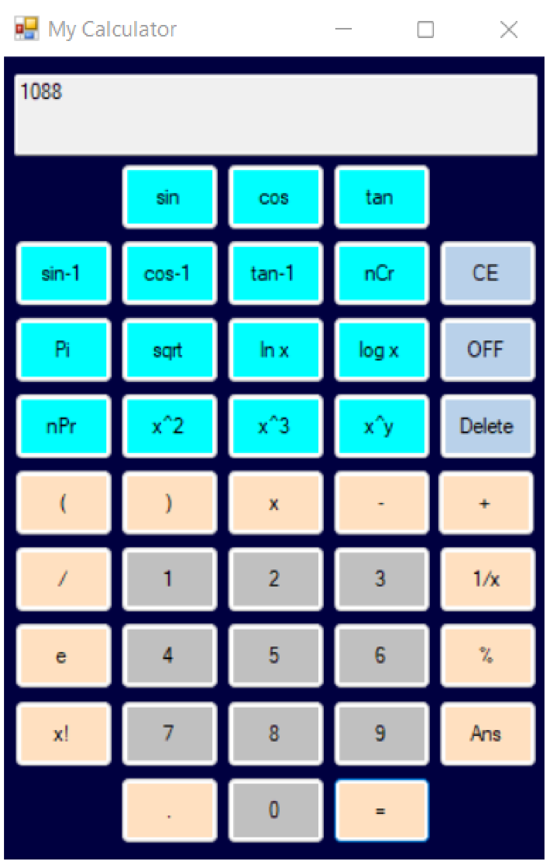


Figure 5: User insert the number and operation symbols



1088

Figure 6: Output (based on figure 5 operation)

Input Validation

According to Figure 7, if a user enters wrong data, an error message will appear in the textbox and they must start over. The error message can appear in a number of different ways. For instance, if the user clicks the equation or answer (=/Ans) button without inserting the second numbers after clicking the operator, an error warning will appear. The operators (%, X2, Inv,, X!, In, log, deg,, x, sin, cos, tan) will receive error messages. If the user clicks the equation (=/Ans) sign without an operator, an error notice may also appear.

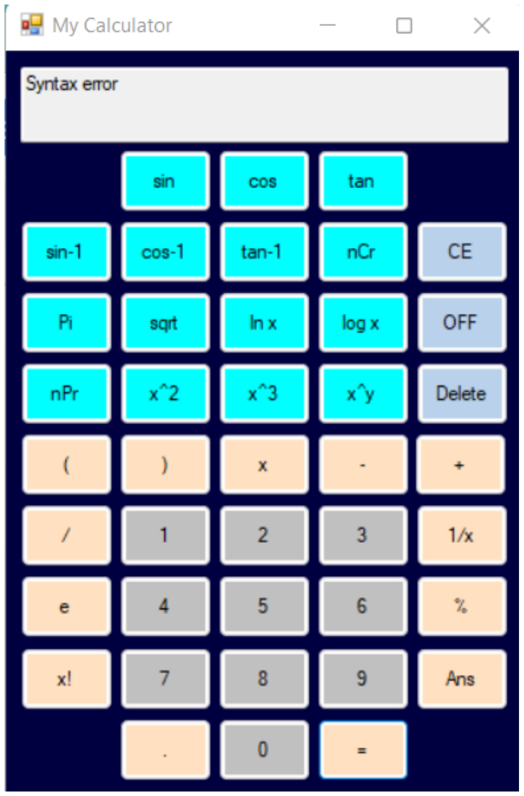


Figure 7: Input Validation