

```
Imports Bestcode.MathParser

Public Class Form1
    Inherits System.Windows.Forms.Form

    Windows Form Designer generated code

    'to measure performance, lets get Win32 help.
    Private Declare Function GetTickCount Lib "kernel32" () As Integer

    Private Sub ButtonCalc_Click(ByVal sender As System.Object, ByVal e As System.
        EventArgs) Handles ButtonCalc.Click
        Try
            Dim myParser As MathParser
            'instantiate a parser:
            myParser = New MathParser

            'You can turn of string support if you want.
            'myParser.StringLiteralsAllowed = False

            'assign a value to X, for example 5.0
            myParser.X = Double.Parse(TextBoxX.Text)

            'assign a value to Y, for example 2.3
            myParser.Y = Double.Parse(TextBoxY.Text)

            'Create a new variable called "k" and assign a handler to it.
            'While evaluating the expression, the parser will ask this
            'delegate for the value of this variable.
            myParser.SetVariable("K", 0.0, New VariableDelegate(AddressOf GetVarValue))

            'Create a new variable called "m" and assign a pre-set value 1.0 to it.
            myParser.SetVariable("M", 1.0, Nothing)

            'assign an expression to parse,
            'for example x+y+sin(x+1)
            myParser.Expression = TextBoxExpression.Text
            myParser.OptimizationOn = True

            'get the value:
            LabelEvalResult.Text = myParser.ValueAsString()

            'Measure repeated evaluation:
            Dim Start As Integer
            Dim i As Int32
            Start = GetTickCount()

            For i = 0 To 1000000
                myParser.Evaluate()
            Next
            Label4.Text = (GetTickCount() - Start) & " milliseconds"

            Catch exception As Exception
                Label4.Text = "n/a"
                System.Console.WriteLine(exception.StackTrace())
                MsgBox(exception.Message)
        End Try
    End Sub

    'Implement function to return variable value:
    Private Function GetVarValue(ByVal sender As MathParser, ByVal varName As String) As
        IConvertible
        If varName = "K" Then 'the parser uses UPPERCASE identifiers.
            GetVarValue = 3.0
        Else
```

```
        'should not happen but just in case:  
        Throw New InvalidExpressionException(varName + " is not a valid variable")  
    End If  
End Function  
  
End Class
```