```
Imports System.Collections.Generic
Imports System.IO
Imports System.Text.RegularExpressions
Imports System.Xml

Imports SMath.Manager
Imports SMath.Math
Imports SMath.Math.Numeric


Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.
    Click

        Dim context As New Store

        Dim termsList As New List(Of List(Of Term))()

        Dim FileName As String = "c:\Мусор\Лист1.sm"

        Try

            Dim doc As New XmlDocument()

            doc.Load(FileName)

            Dim Regions As XmlNodeList = doc.DocumentElement.ChildNodes

            For Each reg In Regions

                Try

                    Dim items As XmlNodeList = reg("math")("input").ChildNodes

                    Dim terms As New List(Of Term)()

                    For Each item As XmlNode In items

                        Dim type As String = item.Attributes("type").Value

                        If type.Equals("operand") Then

                            Dim name As String = item.InnerText

                            Try

                                If item.Attributes("style").Value.Equals("unit") Then
                                    name = "'" + name
                                End If

                                If item.Attributes("style").Value.Equals("string") Then
                                    name = """" + name + """"
                                End If

                            Catch
                            End Try

                            terms.Add(New Term(name, TermType.Operand, 0))

                        ElseIf type.Equals("operator") Then
                            terms.Add(New Term(item.InnerText, TermType.[Operator], Integer.Parse(item.
    Attributes("args").Value)))

                        ElseIf type.Equals("function") Then
                            terms.Add(New Term(item.InnerText, TermType.[Function], Integer.Parse(item.
    Attributes("args").Value)))

                        ElseIf type.Equals("bracket") Then
                            terms.Add(New Term(item.InnerText, TermType.Bracket, 0))

                        End If

                    Next
```

```
                termsList.Add(terms)

            Catch ex As Exception
            End Try

        Next

        For Each terms In termsList

            Try

                Dim ExpressionItems As SMath.Math.Symbolic.MItem = SMath.Math.Symbolic.Converter.ToMItem ↙
(terms.ToArray())
                Dim ExpressionString As String = SMath.Math.Symbolic.Converter.ToString(ExpressionItems)
                Dim pattern As New Regex([String].Format("(\w+|'\w+)(\{0}\w+)*(?=:)", GlobalParams.      ↙
DecimalSymbolStandard))
                Dim m As Match = pattern.Match(ExpressionString)

                If m.Success Then

                    Dim name As String = ExpressionString.Substring(m.Index, m.Length)

                    ExpressionItems = SMath.Math.Symbolic.Converter.ToMItem(ExpressionString.Substring(m.↙
Index + m.Length + 1))
                    context.AddDefinition(name, ExpressionItems.ToTerms(), New Term() {})

                End If

                pattern = New Regex([String].Format("\w+({1}\w+)*(?=\(\w+({1}\w+)*({0}\w+({1}\w+)*)*\):)"↙
, GlobalParams.ArgumentsSeparatorStandard, GlobalParams.DecimalSymbolStandard))
                m = pattern.Match(ExpressionString)

                If m.Success Then

                    Dim name As String = ExpressionString.Substring(m.Index, m.Length)

                    pattern = New Regex([String].Format("(?<=\w+({1}\w+)*\()\w+({1}\w+)*({0}\w+({1}\w+)*)↙
*(?=\):)", GlobalParams.ArgumentsSeparatorStandard, GlobalParams.DecimalSymbolStandard))
                    m = pattern.Match(ExpressionString)

                    Dim parametrs As String() = ExpressionString.Substring(m.Index, m.Length).Split      ↙
(GlobalParams.ArgumentsSeparatorStandard)
                    Dim vars As New List(Of Term)()

                    For Each s In parametrs
                        vars.Add(New Term(s, TermType.Operand, 0))
                    Next

                    ExpressionItems = SMath.Math.Symbolic.Converter.ToMItem(ExpressionString.Substring(m.↙
Index + m.Length + 2))

                    context.AddDefinition(name, ExpressionItems.ToTerms(), vars.ToArray())

                End If

            Catch ex As Exception

            End Try
        Next

    Catch ex As Exception

    End Try

    ' TODO: Initialize SMath Studio engine

    Dim res As Term() = Decision.Preprocessing(New Term() {New Term("c", TermType.Operand, 0)}, context)

    Dim c As TNumber = Decision.NumericCalculation(res, context)

    TextBox1.Text = c.obj.ToDouble().ToString()

End Sub
```

```
End Class
```