Sortierprogramm

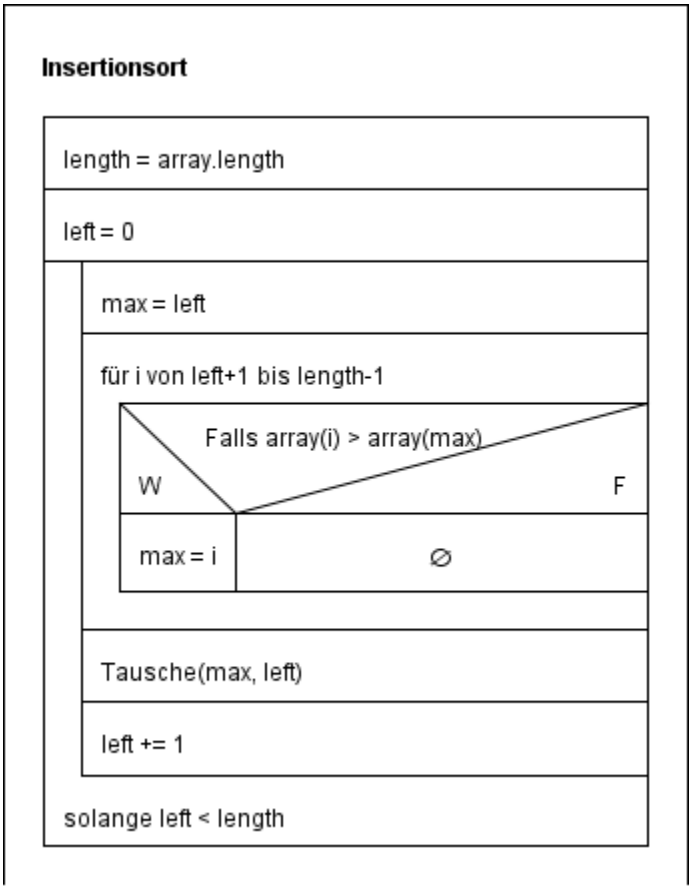
Entwicklerdokumentation Sortierprogramm

Programmbeschreibung

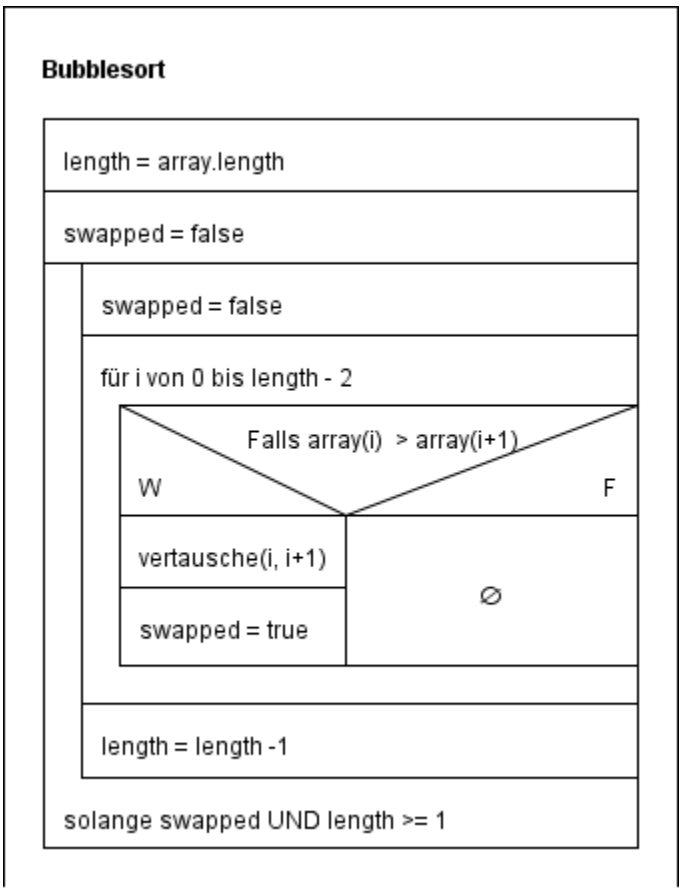
Dieses Programm dient zum Sortieren von Text. Es können diverse Sortieralgorithmen und -optionen ausgewählt werden. Beide Texte (der sortierte und der unsortierte) können als Textdatei gespeichert werden. Unsortierten Text kann aus einer Datei ausgelesen werden.

Sortieralgorithmen - Struktogramme

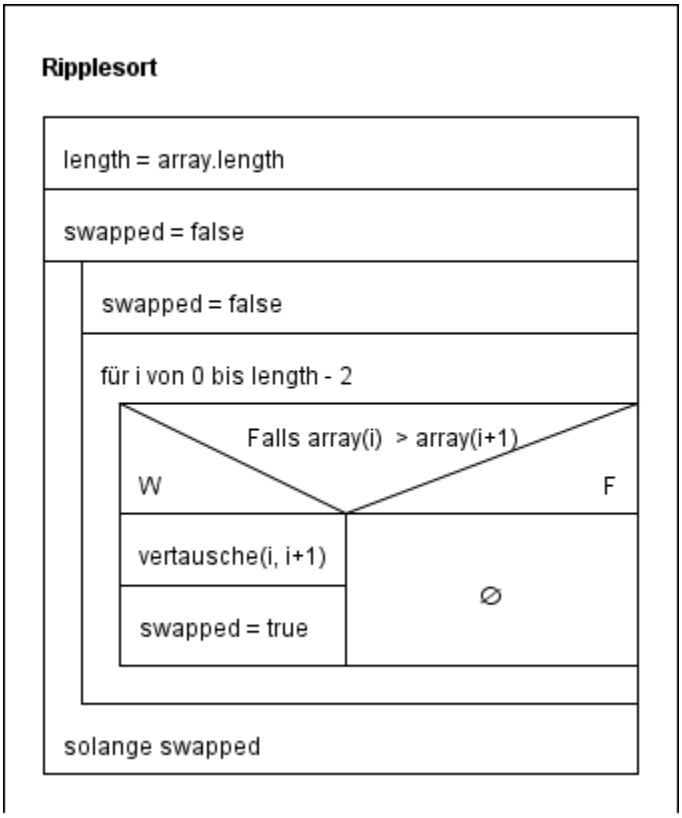
Insertionsort



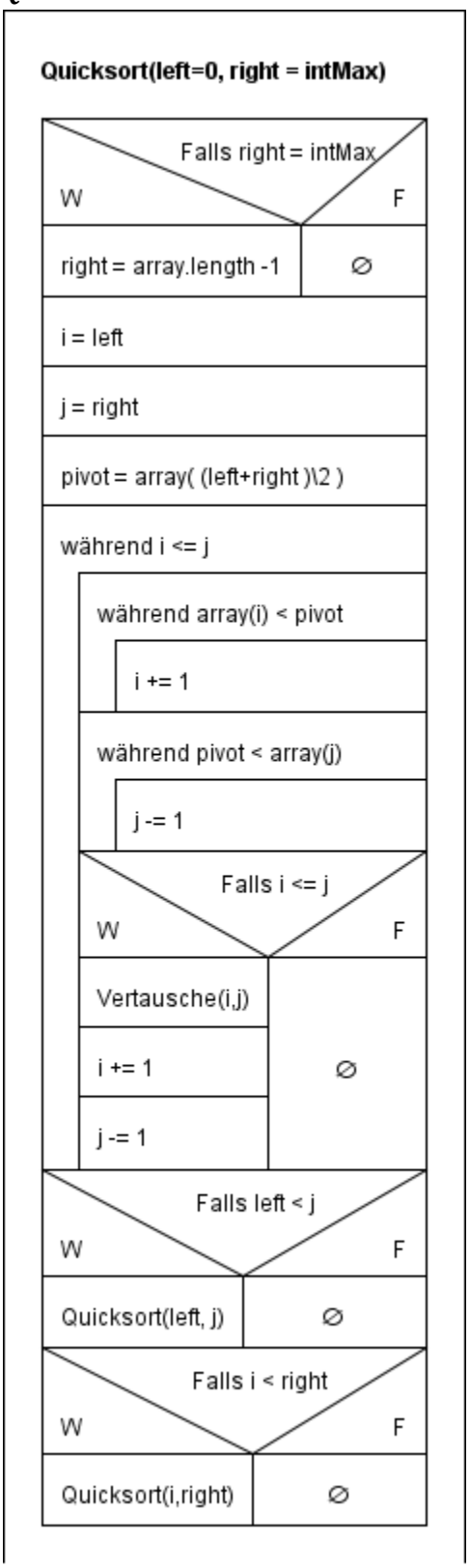
Bubblesort


Bubblesort
length = array.length
swapped = false
swapped = false
für i von 0 bis length - 2
Falls array(i) > array(i+1)
W
F
vertausche(i, i+1)
swapped = true
∅
length = length -1
solange swapped UND length >= 1

Ripplesort


Ripplesort
length = array.length
swapped = false
swapped = false
für i von 0 bis length - 2
Falls array(i) > array(i+1)
W
F
vertausche(i, i+1)
swapped = true
∅
solange swapped

Quicksort


Quicksort(left=0, right = intMax)
Falls right = intMax
W
F
right = array.length -1
∅
i = left
j = right
pivot = array((left+right)\2)
während i <= j
während array(i) < pivot
i += 1
während pivot < array(j)
j -= 1
Falls i <= j
W
F
Vertausche(i,j)
i += 1
j -= 1
∅
Falls left < j
W
F
Quicksort(left, j)
∅
Falls i < right
W
F
Quicksort(i,right)
∅

Module

Modul: frmMain

Variablen

Am Anfang des Codes sind alle Formularweiten Variablen deklariert:

- `blnSortWords`: Speichert ob nach Wörtern sortiert werden soll
- `blnCompareText`: Speichert ob mit CompareMethod.Text oder Binary verglichen werden soll
- `blnUmlaute`: Speichert ob Umlaute als Nicht-Umlaut-Buchstaben verglichen werden sollen.
- `udtSortContainer`: Eine Instanz von meiner eigenen Klasse StringSorting.vb

Subroutinen und Funktionen

Die Subroutinen und Funktionen sind nach Zweck gruppiert. Zuerst die Loadfunktion, dann alle Klick- und andere Handler. Am Schluss die Hilfsfunktionen.

Load

- `frmMain_Load(sender As Object, e As EventArgs) Handles MyBase.Load`: Setzt Startwerte für die Gui.

Handlers

- `btnQuit_Click(sender As Object, e As EventArgs) Handles btnQuit.Click , mnuExit.Click`: Führt die Kill-Subroutine in SortContainer aus. Schliesst das Fenster.
- `btnSort_Click(sender As Object, e As EventArgs) Handles btnSort.Click , mnuSortieren.Click, tlbSort.Click`: Sortiert den Input falls ein Algorithmus gewählt ist, fragt nach einem Algorithmus falls nicht.
- `txtUnsorted_KeyDown(sender As Object, e As KeyEventArgs) Handles txtUnsorted.KeyDown`: Ctrl-A markiert alles im unsortierten Text.
- `NeuToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles mnuNew.Click, tlbNew.Click`: Führt die Kill-Subroutine in SortContainer aus, setzt den sortierten und den unsortieren Text zurück.
- `WorteToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles mnuWorte.Click, chkWords.Click`: Wechselt zwischen Wortsortierung und Buchstabensortierung. Wechselt blnSortWords, setzt GUI-Elemente.
- `chkUmlaute_Click(sender As Object, e As EventArgs) Handles chkCompareText.Click, mnuCompareText.Click`: Wechselt zwischen CompareMethod.Text und CompareMethod.Binary. Wechselt blnCompareText, setzt GUI-Elemente.
- `cmbAlgorithm_SelectedIndexChanged(sender As Object, e As EventArgs) Handles cmbAlgorithm.SelectedIndexChanged`: Wählt einen Algorithmus mit der Hilfsfunktion setAlgorithm.
- `Menu_Algorithm_Click(sender As Object, e As EventArgs) Handles mnuBubblesort.Click, mnuAlgorithmus.Click, mnuInsertionsort.Click, mnuQuicksort.Click, mnuRipplesort.Click`: Wählt einen Algorithmus im Menu und setzt ihn mit der Hilfsfunktion setAlgorithm.

- `mnuOpen_Click(sender As Object, e As EventArgs) Handles mnuOpen.Click, tlbOpen.Click`: Öffnet den Datei-Öffnen-Dialog um einen unsortierten Text einzulesen.
- `ofdLoadText_FileOk(sender As Object, e As System.ComponentModel.CancelEventArgs) Handles dlgLoadText.FileOk`: Liest den Inhalt der gewählten Datei ein.
- `mnuCopy_Click(sender As Object, e As EventArgs) Handles mnuCopy.Click, tlbCopy.Click`: Kopiert den Text via Menu(inkl. Shortcut) oder Toolbar.
- `mnuCut_Click(sender As Object, e As EventArgs) Handles mnuCut.Click, tlbCut.Click`: Schneidet den ausgewählten Text aus via Menu(inkl. Shortcut) oder Toolbar.
- `mnuPaste_Click(sender As Object, e As EventArgs) Handles mnuPaste.Click, tlbPaste.Click`: Fügt den Inhalt des Clipboards ein via Menu(inkl. Shortcut) oder Toolbar.
- `mnuSave_Click(sender As Object, e As EventArgs) Handles mnuSave.Click, tlbSave.Click`: Speichert den Text der momentan aktiven Textbox in einem File. Öffnet einen FileSave-Dialog.
- `show_hide_Items_Handler(sender As Object, e As EventArgs) Handles mnuEdit.DropDownOpening, txtUnsorted.Click, txtSorted.Click, chkWords.Click, chkCompareText.Click, chkUmlaute.Click, btnSort.Click`: versteckt oder Zeigt Menü und Toolbar Elemente je nachdem ob es möglich ist, mit ihnen etwas zu machen.
- mnuAbout_Click(sender As Object, e As EventArgs) Handles mnuAbout.Click, tlbHelp.Click: Zeigt die About-Box.

Hilfsfunktionen

- `sort()`: Übergibt den Text der Klasse StringSorting. In Abhängigkeit der gewählten Optionen wird der korrekte Sortieralgorithmus angewendet. Nach dem Sortieren wird die Anzahl Elemente und die benötigte Zeit angezeigt.
- `textCompare() As Boolean`: Gibt den Wert von blnCompareText zurück.
- `maskUmlaute() As Boolean`: Gibt den Wert von blnUmlaute zurück.
- `setAlgorithm(ByVal strAlgo As String)`: Setzt den in der ComboBox angezeigten sowie den im Menü ausgewählten algorithmus auf strAlgo.

Module: StringSorting

Variablen

Am Anfang des Codes sind alle Formularweiten Variablen deklariert:

- `blnuseWords`: Speichert ob diese Instanz nach Wörtern sortiert oder nicht.
- `strWords()`: Array der Wörter die Sortiert werden müssen
- `blnAlive`: Wenn Falsch werden die Subroutinen abbrechen.
- `alstList`: ArrayList zur einfachen Speicherung der Daten.

Subroutinen und Funktionen

Zuoberst ist der Konstruktor, dann die Sortierfunktionen, dann die Hilfsfunktionen.

Konstruktor

- `New(ByVal strText As String, Optional ByVal blnWords As Boolean = False)`: Liest den text aus strText aus, falls blnWords gesetzt ist, wird der Text bei allen Zeilenumbrüchen und Leerzeichen getrennt, sonst warden alle Zeichen einzeln eingelesen.

Sortierfunktionen

Alle Sortierfunktionen benützen die Funktion strComp zum Vergleichen der Strings. Bei dieser kann man die CompareMethod als Argument mitgeben.

- `Insertionsort()`: Sortiert den Inhalt von strWords mit dem Insertionsort-Algorithmus.
- `BubbleSort()`: Sortiert den Inhalt von strWords mit dem BubbleSort-Algorithmus.
- `RippleSort ()`: Sortiert den Inhalt von strWords mit dem RippleSort-Algorithmus.
- `Quicksort(Optional ByVal intLeft As Integer = 0, Optional ByVal intRight As Integer = Int32.MaxValue)`: Sortiert den Inhalt von strWords mit dem QuickSort-Algorithmus. Falls inLeft und intRight nicht gesetzt sind, werden sie auf 0 bzw. die Länge des Textes gesetzt.

Hilfsfunktionen

- `SwapValues(ByVal intFirst As Integer, ByVal intSecond As Integer)`: Vertauscht die Beiden Wörter (oder Zeichen) die an Position intFirst und intSecond sind.
- `maskUmlaute(ByVal strWord As String) As String`: Gibt einen String zurück, bei dem alle Umlaute durch ihre entsprechenden normalen Buchstaben ersetzt wurden (z.B: ä -> a). Falls frmMain.maskUmlaute() nicht wahr ist, wird der String einfach zurückggeben.
- `ToString() As String`: Gibt den in der KLasse gespeicherten Text zurück. Falls Wörter sortiert wurden, wird ein Leerzeichen zwischen jedes eingefügt, sonst werden alle Zeichen der Reihe nach ausgegeben.
- `compareMethod() As CompareMethod`: Gibt CompareMethod.Text oder CompareMethod.Binary zurück, je nachdem ob frmMain.textCompre() wahr oder falsch ist.
- `kill()`: Setzt blnAlive auf False.
- `Length() As Integer`: Gibt die Länge der sortierten Elemente zurück.

Modul: AboutBox1

Subroutinen und Funktionen

- `AboutBox1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load`: Setzt die Info der AboutBox.
- `OKButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles OKButton.Click`: Schliesst das Fenster bei einem Klick.

Anhang

frmMain.vb

```
Public Class frmMain
    'Variables
    Private blnSortWords As Boolean = False
    Private blnCompareText As Boolean = False
    Private blnUmlaute As Boolean = False
```

```
    Private udtSortContainer As StringSorting = New StringSorting("")

    'Load
    Private Sub frmMain_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        mnuSortieren.ShortcutKeys = Keys.Control Or Keys.Enter ' Not possible via GUI-
Designer
        chkWords.Checked = blnSortWords
        mnuWorte.Checked = blnSortWords
        chkCompareText.Checked = blnCompareText

    End Sub

    'Click and other Handlers
    Private Sub btnQuit_Click(sender As Object, e As EventArgs) Handles btnQuit.Click,
mnuExit.Click
        udtSortContainer.kill()
        Me.Close()
    End Sub


    Private Sub btnSort_Click(sender As Object, e As EventArgs) Handles btnSort.Click,
mnuSortieren.Click, tlbSort.Click
        If cmbAlgorithm.SelectedItem = "" Then
            MsgBox("Please select an algorithm")
        Else
            sort()
        End If
    End Sub

    Private Sub txtUnsorted_KeyDown(sender As Object, e As KeyEventArgs) Handles
txtUnsorted.KeyDown
        If e.Modifiers = Keys.Control AndAlso e.KeyCode = Keys.A Then
            txtUnsorted.SelectAll()
        End If
    End Sub

    Private Sub NeuToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
mnuNew.Click, tlbNew.Click
        udtSortContainer.kill()
        txtSorted.Text = ""
        txtUnsorted.Text = ""
    End Sub

    Private Sub WorteToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
mnuWorte.Click, chkWords.Click
        blnSortWords = Not blnSortWords
        mnuWorte.Checked = blnSortWords
        chkWords.Checked = blnSortWords
    End Sub

    Private Sub chkUmlaute_Click(sender As Object, e As EventArgs) Handles
chkCompareText.Click, mnuCompareText.Click
        blnCompareText = Not blnCompareText
        mnuCompareText.Checked = blnCompareText
        chkCompareText.Checked = blnCompareText
    End Sub
```

```
    Private Sub mnuUmlaute_Click(sender As Object, e As EventArgs) Handles
mnuUmlaute.Click, chkUmlaute.Click
        blnUmlaute = Not blnUmlaute
        mnuUmlaute.Checked = blnUmlaute
        chkUmlaute.Checked = blnUmlaute
    End Sub

    Private Sub cmbAlgorithm_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles cmbAlgorithm.SelectedIndexChanged
        setAlgorithm(cmbAlgorithm.SelectedItem)
    End Sub

    Private Sub Menu_Algorithm_Click(sender As Object, e As EventArgs) Handles
mnuBubblesort.Click, mnuAlgorithmus.Click, mnuInsertionsort.Click, mnuQuicksort.Click,
mnuRipplesort.Click
        Dim item As ToolStripMenuItem = sender
        setAlgorithm(item.Text.Substring(1))
    End Sub

    Private Sub mnuOpen_Click(sender As Object, e As EventArgs) Handles mnuOpen.Click,
tlbOpen.Click
        dlgLoadText.DefaultExt = ".txt"
        dlgLoadText.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"
        dlgLoadText.ShowDialog()

    End Sub

    Private Sub ofdLoadText_FileOk(sender As Object, e As
System.ComponentModel.CancelEventArgs) Handles dlgLoadText.FileOk
        Try
            Using sr As New IO.StreamReader(dlgLoadText.FileName)
                txtUnsorted.Text = sr.ReadToEnd()
            End Using
        Catch
            txtUnsorted.Text = "Could not read the file"
        End Try


    End Sub


    Private Sub mnuCopy_Click(sender As Object, e As EventArgs) Handles mnuCopy.Click,
tlbCopy.Click
        If TypeOf Me.ActiveControl Is TextBox Then
            CType(Me.ActiveControl, TextBox).Copy()
        End If
    End Sub
    Private Sub mnuCut_Click(sender As Object, e As EventArgs) Handles mnuCut.Click,
tlbCut.Click
        If TypeOf Me.ActiveControl Is TextBox Then
            CType(Me.ActiveControl, TextBox).Cut()
        End If
    End Sub

    Private Sub mnuPaste_Click(sender As Object, e As EventArgs) Handles mnuPaste.Click,
tlbPaste.Click
        If TypeOf Me.ActiveControl Is TextBox And Not CType(Me.ActiveControl,
TextBox).ReadOnly Then
```

```
            CType(Me.ActiveControl, TextBox).Paste()
        End If

    End Sub



    Private Sub mnuSave_Click(sender As Object, e As EventArgs) Handles mnuSave.Click,
tlbSave.Click
        Dim myStream As IO.Stream
        dlgSaveText.DefaultExt = ".txt"
        dlgSaveText.AddExtension = True
        dlgSaveText.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"
        dlgSaveText.FilterIndex = 2
        dlgSaveText.RestoreDirectory = True

        If dlgSaveText.ShowDialog() = DialogResult.OK Then
            myStream = dlgSaveText.OpenFile()
            If (myStream IsNot Nothing) Then
                Using writer As IO.StreamWriter = New IO.StreamWriter(myStream,
System.Text.Encoding.Unicode)
                    If TypeOf Me.ActiveControl Is TextBox Then
                        writer.WriteLine(CType(Me.ActiveControl, TextBox).Text)
                    End If
                End Using
                myStream.Close()
            End If
        End If
    End Sub

    Private Sub show_hide_Items_Handler(sender As Object, e As EventArgs) Handles
mnuEdit.DropDownOpening, txtUnsorted.Click, txtSorted.Click, chkWords.Click,
chkCompareText.Click, chkUmlaute.Click, btnSort.Click
        If TypeOf Me.ActiveControl Is TextBox Then
            mnuCut.Enabled = True
            mnuCopy.Enabled = True
            tlbCopy.Enabled = True
            tlbCut.Enabled = True
        Else
            mnuCut.Enabled = False
            mnuCopy.Enabled = False
            tlbCopy.Enabled = False
            tlbCut.Enabled = False
        End If
        If Clipboard.GetDataObject.GetData(DataFormats.Text) <> "" And TypeOf
Me.ActiveControl Is TextBox Then
            If CType(Me.ActiveControl, TextBox).ReadOnly Then
                mnuPaste.Enabled = False
                tlbPaste.Enabled = False
            Else
                mnuPaste.Enabled = True
                tlbPaste.Enabled = True
            End If
        Else
            mnuPaste.Enabled = False
            tlbPaste.Enabled = False
        End If
    End Sub
```

```
    Private Sub mnuAbout_Click(sender As Object, e As EventArgs) Handles mnuAbout.Click,
tlbHelp.Click
        AboutBox1.Show()
    End Sub


    ' Utility Functions


    Private Sub sort()
        txtSorted.Text = ""
        Me.Cursor = Cursors.WaitCursor
        If chkWords.CheckState = 1 Then
            udtSortContainer = New StringSorting(txtUnsorted.Text, True)
        Else
            udtSortContainer = New StringSorting(txtUnsorted.Text)
        End If
        Dim startTime As DateTime = Now
        Select Case cmbAlgorithm.SelectedItem
            Case "Bubblesort"
                udtSortContainer.BubbleSort()
            Case "Insertionsort"
                udtSortContainer.Insertionsort()
            Case "Quicksort"
                udtSortContainer.Quicksort()
            Case "Ripplesort"
                udtSortContainer.RippleSort()
        End Select
        txtDuration.Text = Now.Subtract(startTime).ToString()
        txtWordNum.Text = udtSortContainer.Length()
        txtSorted.Text = udtSortContainer.ToString()
        Me.Cursor = Cursors.Arrow
    End Sub




    Public Function textCompare() As Boolean
        Return blnCompareText
    End Function


    Public Function maskUmlaute() As Boolean
        Return blnUmlaute
    End Function



    Private Sub setAlgorithm(ByVal strAlgo As String)
        cmbAlgorithm.SelectedItem = strAlgo
        For Each s As ToolStripMenuItem In mnuAlgorithmus.DropDownItems
            s.Checked = False
            'Remove the preceding & from the Text
            If s.Text.Substring(1) = strAlgo Then
                s.Checked = True
            End If
```

```
        Next
    End Sub


End Class
```

StringSorting.vb

```
Public Class StringSorting
    Private blnUseWords As Boolean '
    Private strWords As String() ' The objects to be sorted
    Private blnAlive As Boolean = True ' This needs to be here, it doesn't work when
saved in the form. Makes Functions abort when false
    Private alstList As ArrayList


    'Constructor
    Public Sub New(ByVal strText As String, Optional ByVal blnWords As Boolean = False)
        Me.blnUseWords = blnWords
        alstList = New ArrayList
        If Me.blnUseWords Then
            Dim strLines As String() = strText.Split(vbCrLf)
            ' Split each line into words
            For i = 0 To strLines.Length - 1
                Dim strArray = strLines(i).Split(" ")
                ' Add all words to the list
                For j = 0 To strArray.Length - 1
                    alstList.Add(strArray(j))
                Next
            Next
        Else
            For i = 0 To strText.Length - 1 Step 1
                alstList.Add(CStr(strText(i)))
            Next
        End If
        ReDim Preserve strWords(alstList.ToArray.Length - 1)
        alstList.ToArray.CopyTo(strWords, 0)
    End Sub


       'Sorting Functions

    Public Sub Insertionsort()
        Dim intLength As Integer = strWords.Length
        Dim intLeft As Integer = 0
        Do
            Dim intMax As Integer = intLeft
            For i = intLeft + 1 To intLength - 1
                If StrComp(maskUmlaute(strWords(i)), maskUmlaute(strWords(intMax)),
compareMethod()) = -1 Then
                    intMax = i
                End If
                If Not blnAlive Then ' Make it correctly quit when quitting :)
                    Exit Sub
                End If
            Next
```

```
            SwapValues(intMax, intLeft)
            intLeft += 1
        Loop While intLeft < intLength
        alstList.Clear()
        alstList.AddRange(strWords)
    End Sub

    Public Sub BubbleSort()
        Dim length As Integer = strWords.Length
        Dim swapped As Boolean = False
        Do
            swapped = False
            For i = 0 To length - 2
                If StrComp(maskUmlaute(strWords(i)), maskUmlaute(strWords(i + 1)),
compareMethod()) = 1 Then
                    SwapValues(i, i + 1)
                    swapped = True
                End If
                If Not blnAlive Then ' Make it correctly quit when quitting :)
                    Exit Sub
                End If
            Next
            length = length - 1
        Loop While swapped And length >= 1

    End Sub

    Public Sub RippleSort()
        Dim length As Integer = strWords.Length
        Dim swapped As Boolean = False
        Do
            swapped = False
            For i = 0 To length - 2
                If StrComp(maskUmlaute(strWords(i)), maskUmlaute(strWords(i + 1)),
compareMethod()) = 1 Then
                    SwapValues(i, i + 1)
                    swapped = True
                End If
                If Not blnAlive Then ' Make it correctly quit when quitting :)
                    Exit Sub
                End If
            Next
        Loop While swapped

    End Sub


    Public Sub Quicksort(Optional ByVal intLeft As Integer = 0, Optional ByVal intRight
As Integer = Int32.MaxValue)
        If intRight = Int32.MaxValue Then
            intRight = strWords.Length - 1 'index of the right most element
        End If
        Dim i = intLeft
        Dim j = intRight
        Dim x As String = strWords((intLeft + intRight) \ 2)
        While i <= j

            While StrComp(maskUmlaute(strWords(i)), maskUmlaute(x), compareMethod()) = -1
```

```
                i += 1
            End While
            While StrComp(maskUmlaute(x), maskUmlaute(strWords(j)), compareMethod()) = -1
                j -= 1
            End While
            If i <= j Then
                SwapValues(i, j)
                i += 1
                j -= 1

                If Not blnAlive Then ' Make it correctly quit when quitting :)
                    Exit Sub
                End If
            End If
        End While

        If intLeft < j Then
            Quicksort(intLeft, j)
        End If
        If i < intRight Then
            Quicksort(i, intRight)
        End If

    End Sub

    'Utility Functions

    Private Sub SwapValues(ByVal intFirst As Integer, ByVal intSecond As Integer)
        Application.DoEvents() ' Make the program responsive (in here because all sort-
functions use this)
        Dim temp As String
        temp = strWords(intFirst)
        strWords(intFirst) = strWords(intSecond)
        strWords(intSecond) = temp
    End Sub


    Private Function maskUmlaute(ByVal strWord As String) As String
        If frmMain.maskUmlaute() Then
            For i As Integer = 1 To strWord.Length
                Select Case strWord(i - 1)
                    Case "ä"
                        Mid(strWord, i) = "a"
                    Case "à"
                        Mid(strWord, i) = "a"
                    Case "ö"
                        Mid(strWord, i) = "o"
                    Case "ü"
                        Mid(strWord, i) = "u"
                    Case "é"
                        Mid(strWord, i) = "e"
                    Case "è"
                        Mid(strWord, i) = "e"
                    Case "ç"
                        Mid(strWord, i) = "c"

                    Case "Ä"
                        Mid(strWord, i) = "A"
```

```
                    Case "À"
                        Mid(strWord, i) = "A"
                    Case "Ö"
                        Mid(strWord, i) = "O"
                    Case "Ü"
                        Mid(strWord, i) = "U"
                    Case "É"
                        Mid(strWord, i) = "E"
                    Case "È"
                        Mid(strWord, i) = "E"
                End Select
            Next
        End If
        Return strWord
    End Function



    Public Overrides Function ToString() As String
        alstList.Clear()
        alstList.AddRange(strWords)
        Dim strReturn As String = ""
        Dim e As IEnumerator
        e = alstList.GetEnumerator()
        While (e.MoveNext())
            Dim objWord As Object = e.Current
            strReturn &= CStr(objWord)
            If Me.blnUseWords Then
                strReturn &= " "
            End If
        End While
        Return strReturn
    End Function


    Private Function compareMethod() As CompareMethod
        If frmMain.textCompare() Then
            Return CompareMethod.Text
        Else
            Return CompareMethod.Binary
        End If
    End Function

       Public Sub kill()
        blnAlive = False
    End Sub

    Public Function Length() As Integer
        Return alstList.ToArray.Length
    End Function

End Class
```

AboutBox1.vb

```
Public NotInheritable Class AboutBox1
```

```
    Private Sub AboutBox1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        ' Set the title of the form.
        Dim ApplicationTitle As String
        If My.Application.Info.Title <> "" Then
            ApplicationTitle = My.Application.Info.Title
        Else
            ApplicationTitle =
System.IO.Path.GetFileNameWithoutExtension(My.Application.Info.AssemblyName)
        End If
        Me.Text = String.Format("About {0}", ApplicationTitle)
        ' Initialize all of the text displayed on the About Box.
        Me.LabelProductName.Text = My.Application.Info.ProductName
        Me.LabelVersion.Text = String.Format("Version {0}",
My.Application.Info.Version.ToString)
        Me.LabelCopyright.Text = My.Application.Info.Copyright
        Me.LabelCompanyName.Text = My.Application.Info.CompanyName
        Me.TextBoxDescription.Text = My.Application.Info.Description
    End Sub

    Private Sub OKButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles OKButton.Click
        Me.Close()
    End Sub

End Class
```