7.4.9.1 Sortierung eindimensionaler Arrays

7.4.9.1.1 Beispiel 1 – Sortierung eines eindimensionalen nativen Arrays

Das Sortieren der Elemente von eindimensionalen nativen Arrays funktioniert mit der Sort(..)-Methode problemlos – wie das folgende Beispiel zeigt:

```
Public Sub btnNativesArraySortieren_Click()
  Dim iCount, k As Integer
  Dim a1DArray As New String[]
  Dim aNames As String[] = ["Adler", "Bär", "Dachs", "Fuchs", "Meise", "Uhu", "Elch"]

  iCount = 5
  a1DArray.Resize(iCount)

  Randomize

  For k = 0 To a1DArray.Bounds[0] - 1
    a1DArray[k] = aNames[Int(Rnd(0, aNames.Count))]
  Next ' k

  ShowElements(a1DArray)
    a1DArray.Sort(gb.Ascent) ' Sortier-Standard a..z
  ShowElements(a1DArray)
    a1DArray.Reverse()
  ShowElements(a1DArray)

End ' btnNativesArraySortieren_Click()

Public Sub ShowElements(aArray As String[])
  Dim sElement As String

  For Each sElement In aArray
    Print sElement,
  Next ' sElement
  Print

End ' ShowElements(aArray As String[])
```

Anzeige der Elemente des Arrays *a2DArray* in der Konsole der Gambas-IDE:

```
Bär     Uhu     Adler   Meise   Fuchs   → Original
Adler   Bär     Fuchs   Meise   Uhu     → Aufsteigend sortiert
Uhu     Meise   Fuchs   Bär     Adler   → Sortierte Elemente invertiert sortiert
```

Hinweis:

Das Umsortieren der Elemente in einem Array mit der Methode *Array.Reverse()* kann als Spezialfall des Sortierens eines Arrays aufgefasst werden. Dabei werden alle Elemente von [E0 bis Ek] im Array auf [Ek bis E0] umsortiert.

7.4.9.1.2 Beispiel 2 – Sortierung eines eindimensionalen abgeleiteten Arrays

Es sollen die Namen aller Steuerelemente vom Typ 'Label' auf dem Formular erkundet und sortiert ausgegeben werden. Die folgende Idee zur Umsetzung klingt plausibel:

- Zuerst werden alle Steuerelemente vom Typ Label erfasst und die gefundenen Label als Objekt in einem *abgeleiteten Array* aLabels vom Typ Label[ ] gespeichert.
- Dann wird das Array aLabels (aufsteigend) sortiert.
- Anschließend werden die Label-Namen der sortierten Label im Array aLabels ausgelesen und angezeigt.

Folgender Quelltext wird eingesetzt:

```
Public Sub btnSortLabel1_Click()
  Dim k As Integer
  Dim lLabel As Label
  Dim aLabels As New Label[]
  Dim hControl As Control

  For Each hControl In Me.Children
    If Object.Type(hControl) = "Label" Then
```

```
      Inc k
      aLabels.Resize(k)
      aLabels[k - 1] = hControl
    Endif
  Next ' hControl

  For Each lLabel In aLabels ' Zur Kontrolle
    Print lLabel.Name
  Next
  aLabels.Sort(gb.Ascent)
  For Each lLabel In aLabels
    Print lLabel.Name
  Next

End ' btnSortLabel1_Click()
```

Zur Kontrolle werden zuerst die Namen der Label aus dem *originalen, unsortierten Array* ausgegeben. Es zeigte sich dieses Ergebnis in der Konsole der IDE:

```
lblMultiArray
lblClassArray
lblLabels
lblDemonstration
```

Die Reihenfolge der Elemente im Array aLabels entspricht der Reihenfolge der einzelnen Label in der *Hierarchie* der sichtbaren Steuerelemente in der Gambas-IDE. Mit dem Ergebnis bei der Ausgabe der Namen der Label aus dem sortierten Array aLabels kann man nicht zufrieden sein:

```
lblClassArray
lblLabels
lblMultiArray
lblDemonstration
```

Sortiert sieht anders aus! Der Grund ist darin zu sehen, dass die Elemente in einem Array vom Typ Label[ ] – als einem abgeleitetem Array – nach ihren *Adressen im Speicher* verglichen werden, da die Klasse Label *keine* _compare()-Methode implementiert.

Die ursprüngliche Idee wird aus diesem Grund modifiziert:

- Zuerst werden alle Steuerelemente vom Typ Label erfasst und von allen gefundenen Labeln sofort die *Label-Namen* in einem Array aLabelNames vom Typ String[ ] gespeichert.
- Dann wird das native Array aLabelNames (aufsteigend) sortiert.
- Abschließend werden die Label-Namen aus dem *sortierten* Array aLabelNames ausgelesen und angezeigt.

Der modifizierte Quelltext ist hier nachzulesen:

```
Public Sub btnSortLabel2_Click()
  Dim k As Integer
  Dim sName As String
  Dim aLabelNames As New String[] ' Natives Array
  Dim hControl As Control

  For Each hControl In Me.Children
    If Object.Type(hControl) = "Label" Then
       aLabelNames.Add(hControl.Name)
    Endif
  Next ' hControl
  For Each sName In aLabelNames ' Zur Kontrolle
    Print sName
  Next
  Print
  aLabelNames.Sort(gb.Ascent)
  For Each sName In aLabelNames
    Print sName
  Next
End ' btnSortLabel2_Click()
```

In der Konsole wird nun die korrekte Liste der aufsteigend sortierten Label-Namen angezeigt:

```
lblClassArray
lblDemonstration
lblLabels
lblMultiArray
```