7.4.5 Inline-Array

Mit der Inline-Notation eines Arrays können Sie auf kompakte Weise einerseits ein Array erzeugen und es andererseits sofort mit Daten füllen.

Syntax für ein Inline-Array:

```
Array = [ Expression [ , ... ] ]
```

Die Anweisung erzeugt ein Array und gibt es zurück. Der Daten-Typ *aller* Elemente wird geprüft. Wenn diese alle vom *gleichen* Typ sind oder wenn sie in den gleichen nativen Datentyp *konvertiert* werden können, dann wird ein Array von einem speziellen Typ zurückgegeben. Zum Beispiel String[ ], wenn alle Elemente nur String-Werte sind, Float[ ] bei Fließkomma-Zahlen und so weiter... . Ansonsten wird ein Variant[ ]-Array zurückgegeben. Sie erzeugen ein Array innerhalb eines Ausdrucks mit dem [...]-Operator.

Die nächsten 2 Zeilen sind so zu lesen:

```
Dim aZeichenArray As String[]
aZeichenArray = ["I", "IV", "V", "IX", "X", "XL", "L", "XC", "C", "CD", "D", "CM", "M"]
```

Dem deklarierten Array 'aZeichenArray' wird das *Inline-Array* ["I", "IV", .., "M"] zugewiesen. Es wurde anonym deklariert oder wie es in der Gambas-Dokumentation genannt wird: inline (→ http://gambaswiki.org/wiki/lang/array). Das Inline-Array ["I", "IV", .., "M"] ist ein vollwertiges Array – jedoch ohne Namen und kann wie ein normales Array benutzt werden.

Hinweis:

Das Array ["I", "IV", .., "M"] ist ein Inline-Array - aber kein eingebettetes Array, obwohl beide Begriffe sinngemäß nahezu gleich sind. Das "Inline" bezieht sich auf die Art und Weise, wie man das Array notiert. Bei "Embedded"-Arrays bezieht sich "Embedded" auf die Lage des Arrays im Speicher. Weitere Informationen finden Sie im → Kapitel 7.4.6 Embedded-Array.

7.4.5.1 Beispiel 1

Im ersten Projekt wird ein Inline-Array mit 5 Elementen angegeben, die alle einen *anderen* Daten-Typ besitzen:

```
Dim aArray As Array
Dim vElement As Variant
Dim iCount As Integer

aArray = ["Adam", 44, True, Sqr(9), Year(Now)]
Print "Typ von aArray = "; Object.Type(aArray)

For Each vElement In aArray
  Print vElement
Next

Print "Alternative Ausgabe über einen Index:"
For iCount = 0 To aArray.Max
  Print aArray[iCount]
Next
```

Ausgaben in der IDE-Konsole:

```
Typ von aArray = Variant[]

Adam
44
True
3
2014
```

Die alternative Ausgabe über einen Index liefert das gleiche Ergebnis.

7.4.5.2 Beispiel 2

Auf das Inline-Array ["A", "B", "C"] wird die Join(..)-Methode angewendet:

```
PRINT ["A", "B", "C"].Join("/")
Ausgabe: A/B/C
```

7.4.5.3 Beispiel 3

Die beiden Inline-Arrays werden im folgenden Quelltext eingesetzt:

```
["I", "IV", "V", "IX", "X", "XL", "L", "XC", "C", "CD", "D", "CM", "M"]
[1, 4, 5, 9, 10, 40, 50, 90, 100, 400, 500, 900, 1000]
```

und werden in den Zeilen * und ** den deklarierten Arrays 'aZeichenArray' und 'aZahlenArray' auf *unterschiedliche* Art zugewiesen:

```
Function IntegerZahlToZahlZeichen(iIntegerZahl As Long) As String
  Dim aZeichenArray As String[]
  Dim aZahlenArray As New Integer[]
  Dim iCount As Integer
  Dim sZahlZeichen As String

' Verwendung von Inline-Arrays
  aZeichenArray = ["I", "IV", "V", "IX", "X", "XL", "L", "XC", "C", "CD", "D", "CM", "M"]   *
  aZahlenArray.Insert([1, 4, 5, 9, 10, 40, 50, 90, 100, 400, 500, 900, 1000])              **

  For iCount = 12 To 0 Step -1
    While (iIntegerZahl >= aZahlenArray[iCount])
      iIntegerZahl = iIntegerZahl - aZahlenArray[iCount]
      sZahlZeichen = sZahlZeichen & aZeichenArray[iCount]
    Wend
  Next ' iCount

  Return sZahlZeichen

End ' ZahlToZahlzeichen → Konvertierung nach einem Ansatz von Carlos Alberto Longen
```

7.4.5.4 Beispiel 4

```
Dim a2DArrayNames As New String[10, 5]

Randomize

For i = 0 To a2DArrayNames.Bounds[0] - 1
  For j = 0 To a2DArrayNames.Bounds[1] - 1
    a2DArrayNames[i, j] = ["Adam", "Bruno", "Carina", "Detlef", "Erwin"][Rnd(0, 5)]      ***
  Next ' j
Next ' i
```

Das String-Array der Namen ist:

```
["Adam", "Bruno", "Carina", "Detlef", "Erwin"]
```

Aus diesem Inline-Array wird mit zufälligem Index im Bereich von 0 bis 4(!) ein Element ausgewählt, da die Konvertierung in einen Integer-Wert stets alle Dezimalen abschneidet (trunkiert). Die Zeile *** weist so einem über die i,j-Koordinaten adressierten Element des nativen mehrdimensionalen Arrays 'a2DArrayNames' einen zufälligen Namen zu.

Hier folgt der vollständige Quelltext der Prozedur, aus dem der o.a. adaptierte Quelltext-Abschnitt entnommen wurde:

```
Public Sub btnShow2D_Click()
  Dim i, j, k As Integer
  Dim a2DArrayNames As New String[][]
  Dim myArray As New String[]
```

```
  Randomize

  a2DArray.Resize(10)

  For i = 0 To 9
    a2DArrayNames[i] = New String[5]
    For j = 0 To 4
      a2DArrayNames[i][j] = ["Adam", "Bruno", "Carina", "Detlef", "Erwin"][Rnd(0, 5)]
    Next
  Next

' Strukturierte Ausgabe Konsole:
  For Each myArray In a2DArrayNames
    For j = 0 To myArray.Max
        If (j + 1) Mod (myArray.Max + 1) = 0 Then
           Print myArray[j]
        Else
          Print myArray[j],
        Endif
    Next
  Next

' Alternative
  For Each myArray In a2DArrayNames
    For j = 0 To myArray.Max
      Print myArray[j],
      If (j + 1) Mod (myArray.Max + 1) = 0 Then Print
    Next ' j
  Next ' myArray

End
```

Strukturierte Ausgabe in der IDE-Konsole:

```
Carina  Adam    Adam    Detlef  Detlef
Adam    Erwin   Adam    Bruno   Bruno
Erwin   Erwin   Detlef  Carina  Adam
Carina  Detlef  Adam    Detlef  Carina
Adam    Erwin   Detlef  Bruno   Erwin
Detlef  Detlef  Detlef  Bruno   Erwin
Erwin   Erwin   Bruno   Erwin   Erwin
Erwin   Carina  Erwin   Carina  Detlef
Bruno   Adam    Detlef  Detlef  Bruno
Adam    Detlef  Detlef  Carina  Adam
```

7.4.5.5 Beispiel 5

Die einfachste Art, um Daten aus einem Prozess zu lesen (→ Kapitel 21.3.1 Einsatz Quick-Syntax) , besteht darin, die Kurzform der EXEC-Instruktion – die sogenannte Quick-Syntax – zu verwenden:

```
EXEC  aCommand TO (String-)Variable ' aCommand ist ein (String-)Array
```

```
Public Sub btnPingOverExec_Click()
 Dim sAusgabe As String

  FMain.Mouse = Mouse.Wait

  EXEC [sProgrammName, TextBox1.Text, "-c", "4"] To sAusgabe ' String-Array als Inline-Array

  TextArea.Insert(gb.NewLine & sAusgabe)
  FMain.Mouse = Mouse.Default

End ' btnPingOverExec_Click
```

7.4.5.6 Beispiel 6

Im folgenden Beispiel wird gezeigt, wie der Einsatz von Inline-Arrays die Übersichtlichkeit im Quelltext beträchtlich erhöhen kann. Benötigt werden in den einzelnen Varianten die deutschen Monatsnamen oder die Wochentage für ein bestimmtes Datum.

1. Variante

```
Dim dDate as Date
Dim sAktuellerMonat as String

IF Month(dDate) = 12 THEN sAktuellerMonat = "Dezember"
IF Month(dDate) = 11 THEN sAktuellerMonat = "November"
IF Month(dDate) = 10 THEN sAktuellerMonat = "Oktober"
IF Month(dDate) = 9 THEN sAktuellerMonat = "September"
IF Month(dDate) = 8 THEN sAktuellerMonat = "August"
IF Month(dDate) = 7 THEN sAktuellerMonat = "Juli"
IF Month(dDate) = 6 THEN sAktuellerMonat = "Juni"
IF Month(dDate) = 5 THEN sAktuellerMonat = "Mai"
IF Month(dDate) = 4 THEN sAktuellerMonat = "April"
IF Month(dDate) = 3 THEN sAktuellerMonat = "März"
IF Month(dDate) = 2 THEN sAktuellerMonat = "Februar"
IF Month(dDate) = 1 THEN sAktuellerMonat = "Januar"
```

2. Variante

```
Public Function GetMonth(dDate) As String
  Select Case Month(dDate)
    Case 1
      Return "Januar"
    Case 2
      Return "Februar"
    ...
    Case 12
      Return "Dezember"
  End Select
End

sAktuellerMonat = GetMonth(Now())
```

3. Variante – kurz und bündig:

```
Dim aMonate As String[]

aMonate = ["Januar", "Februar", "März", "April", ..., "Oktober", "November", "Dezember"]
sAktuellerMonat = aMonate[Month(dDate)-1]
```

Es folgt ein analoges Beispiel für die Anzeige des Wochentages in einer TextBox für ein Datum, das über die Komponente *DateChooser* ausgewählt wurde:

```
aWochentage = ["Sonntag", "Montag", "Dienstag", …, "Donnerstag", "Freitag", "Samstag"]
txbDateDayOfWeek.Text = aWochentage[WeekDay(DateChooser1.Value)] ' Sonntag = 0!
```

7.4.5.7 Beispiel 7

Hier wird ein Inline-Array bereits bei der Array-Deklaration eingesetzt:

```
Dim aNames As String[] = ["Adler", "Bär", "Dachs", "Fuchs", "Meise", "Uhu", "Elch"]
```

7.4.5.8 Beispiel 8

Für das Einfügen von Daten in eine Matrix können Sie auch Inline-Arrays verwenden, von den drei im folgenden Quelltext-Ausschnitt benutzt werden:

```
Dim hMatrix As Matrix

hMatrix = New Matrix(3, 5, False) ' Matrix aus 3 Zeilen mit 5 Spalten, reelle Zahlen

' Matrix mit Werten füllen

hMatrix.SetRow(0, [11, 12, 13, 14, 15]) ' Elemente hier nur vom Typ Float oder Complex
hMatrix.SetRow(1, [21, 22, 23, 24, 25])
hMatrix.SetRow(2, [31, 32, 33, 34, 35])
```

7.4.5.9 Beispiel 9

Class.New (gb) instanziiert eine Klasse:

```
Function New ( [ Arguments As Array ] ) As Object
```

Diese Routine funktioniert genau wie der NEW Operator – mit der Ausnahme – dass die Konstruktor-Argumente durch ein *Inline-Array* angegeben werden müssen.