10.3.1 Zählschleife – FOR-Kontroll-Struktur

In diesem Kapitel wird die FOR-Kontroll-Struktur beschrieben. Sie ist *eine* Form der Loop-Kontroll-Strukturen mit fester Wiederholungszahl.

10.3.1.1 Syntax

Syntax für die FOR-Kontroll-Struktur:

```
FOR Variable = Expression { TO | DOWNTO } Expression [ STEP Expression ]
   <Anweisung(en) >
NEXT
```

10.3.1.2 Hinweise zur Syntax

- Wiederholt eine Folge von Anweisungen, wobei der Wert einer (Schleifen-)Variable erhöht (inkrementiert) oder vermindert (dekrementiert) wird.
- Bei dieser Form der Wiederholung einer Anweisung oder Anweisungsfolge sind sowohl die Anzahl der Wiederholungen der <Anweisung(en)> als auch die Schrittweite STEP beim Start der Zählschleife bereits bekannt – entweder fest vorgegeben oder berechnet.
- Die Schleifen-Variable muss zwei Bedingungen erfüllen: Sie muss eine lokale Variable sein und vom Typ: Byte, Short, Integer, Long oder Fließkomma-Zahl (Float).
- Ist die Schrittweite vom Typ Float, dann muss auch die (Schleifen-)Variable diesen Typ besitzen.
- Wird das Schlüsselwort DOWNTO an Stelle von TO benutzt, so wird der entgegengesetzte Wert der Schrittweite verwendet.
- Ist der Wert des Start-Ausdrucks größer als der des Ende-Ausdrucks – bei positiven Werten für die Schrittweite – oder ist der Start-Ausdruck kleiner als der des Ende-Ausdrucks bei negativem Wert für die Schrittweite, dann wird die FOR-Kontroll-Struktur *nicht* ausgeführt.

10.3.1.3 Beispiel 1

In diesen 4 Teil-Beispielen werden unterschiedliche Typen für die (Schleifen-)Variable, verschiedene Schrittweiten und die Schlüsselworte TO sowie DOWNTO verwendet:

```
Public Sub btnForToNext_Click()
  Dim iCount As Integer
  Dim fCount As Float

  For iCount = 1 To 7 Step 1 ' Step 1 kann entfallen → Standard-Schrittweite
    Print iCount;;
  Next
    Print
  For fCount = 1 To 7.3 Step 0.8
    Print fCount;;
  Next
    Print
  For iCount = 12 To 7.3 Step -1 ' Alternative Notation für DOWNTO
    Print iCount;;
  Next
    Print
  For fCount = 5.3 DownTo 1 Step 0.5
    Print fCount;;
  Next

End
```

Das wird in der Konsole der Gambas-IDE angezeigt:

```
1 2 3 4 5 6 7
1 1,8 2,6 3,4 4,2 5 5,8 6,6
12 11 10 9 8 7
5,3 4,8 4,3 3,8 3,3 2,8 2,3 1,8 1,3
```

10.3.1.4 Beispiel 2

Der Inhalt einer Datei wird zeilenweise ausgelesen und in einem Array gespeichert, solange noch Zeilen auslesbar sind. Anschließend wird der Inhalt des Arrays (Test-Option) zur Kontrolle in der Konsole der IDE angezeigt:

```
  hFile = Open sRubrikPfad For Input
  While Not Eof(hFile)
    Line Input #hFile, sZeile
    aSuchDateiMatrix.Add(sZeile)
  Wend
  Close #hFile
  aSuchDateiMatrix.Sort(0)

' Zur Kontrolle:
  FOR k = 0 TO aSuchdateiListe.Max
    PRINT aSuchdateiListe[k]
  NEXT ' k
```

10.3.1.5 Beispiel 3

Mithilfe zweier verschachtelter FOR-Kontroll-Strukturen wird der einfache Sortier-Algorithmus *Selectionsort* (→ Wikipedia: http://de.wikipedia.org/wiki/Selectionsort) für ein Integer-Array realisiert:

```
Private Sub Selectionsort(aArray As Integer[])
  Dim iI, iJ, iMin As Integer

  For iI = 0 To aArray.Max
    iMin = iI
    For iJ = iI To aArray.Max
      If aArray[iJ] < aArray[iMin] Then iMin = iJ
    Next
    Swap aArray[iI], aArray[iMin]
  Next
End
```

10.3.1.6 Beispiel 4

Die Join()-Methode der Klasse String[] verbindet die Elemente des Arrays der Reihe nach durch einen Separator-String – zum Beispiel durch ein Komma. Dies erleichtert u.a. die Ausgabe von Arrays zu Testzwecken. Leider fehlt diese Methode in den übrigen nativen Array-Klassen, wie bei Integer[], obwohl mithilfe der Str$()-Funktion die Repräsentation eines Wertes eines beliebigen Datentyps als String geliefert wird. Mithilfe einer FOR-Kontroll-Struktur kann eine solche *generische Join()-Funktion* implementiert werden:

```
Private Function GenericJoin(aArray As Variant[], sSep As String) As String
  Dim iInd As Integer
  Dim sRes As String

  For iInd = 0 To aArray.Max
    sRes &= Str$(aArray[iInd]) & sSep
  Next
  Return Left$(sRes, - Len(sSep))
End
```

10.3.1.7 Hinweise zur Semantik

Die Ausdrücke im Kopf der FOR-Kontroll-Struktur werden einmalig – zum Eintritt in die Schleife – ausgewertet. Damit unterscheidet sich eine FOR-Kontroll-Struktur von while- oder repeat-artigen Schleifen, welche vor respektive nach jeder Iteration die Ausdrücke in ihren Bedingungen erneut evaluieren.

Vorsicht ist deshalb bei nicht-konstanten Ausdrücken geboten, wie Sie das in den folgenden drei Beispielen erkennen werden:

Beispiel 1

```
Private Sub BrokenInfiniteLoop()
  Dim i As Integer

  For i = 0 To i + 1
    Print i
  Next

End
```

Auf den ersten Blick würden Sie annehmen, dass die Bedingung unerfüllbar sei: *i+1* wird immer größer sein als i. Jedoch wird der Ausdruck *i+1* als Ende-Ausdruck nur einmal am Beginn der Schleife ausgewertet, entspricht also 0 + 1 = 1, da i zu diesem Zeitpunkt gleich 0 ist. Die Prozedur wird nach folgender Ausgabe in der Konsole also tatsächlich beendet:

```
0
1
```

Beispiel 2

```
Private Sub BrokenSqr(n As Integer)
  Dim i As Integer, j As Integer = 1

  For i = 1 To n Step 2 * j - 1
    Inc j
  Next
  Dec j

' Zur Kontrolle:
  Print "j="; j, "sqr(n)="; Int(Sqr(n))
End
```

Diese Routine soll den ganzzahligen Teil der Quadratwurzel der Integer-Variablen n ermitteln und in j speichern. Die Idee dieser Routine liegt in der Identität

$$n^2 = (n-1)^2+2\cdot n-1 = \sum_{i=1}^{n}(2i+1)$$

Allerdings wird hier der Step-Ausdruck nur zu *Beginn* der FOR-Kontroll-Struktur ausgewertet. Da zu diesem Zeitpunkt j = 1 gilt, ist der Step-Ausdruck konstant 2 * 1 – 1 = 1. Es folgt die Ausgabe:

```
j=20   sqr(n)=4
```

Beispiel 3

```
Private Sub BrokenArrayRemove()
  Dim a As Integer[] = [1, 2, 3, 4]
  Dim i As Integer

' Alle ungeraden Zahlen aus dem Array entfernen
  For i = 0 To a.Max
    If a[i] Mod 2 Then
      a.Remove(i)
    ' Index nach Entfernen korrigieren, um kein Element zu überspringen
      Dec i
    Endif
  Next

  Catch
    Print "Error: "; Error.Text;; "with i="; i
End
```

Es sollen alle *ungeraden* Zahlen aus dem Array entfernt werden. Bei der Evaluierung des Ende-Ausdrucks wird a.Max = 3 ermittelt. Daher steht zu diesem Zeitpunkt für den Interpreter bereits fest, dass die Schleife exakt 3 – i + 1 = 4-mal auszuführen ist, da i = 0. Allerdings wird in der ersten Iteration a[i] = a[0] = 1 als ungerade erkannt und entfernt. So ergeht es auch der 3 bei i = 1. Danach ist a.Max = 1, die Schleife wird ausgeführt bis einschließlich i = 3, da dieser Wert vom Interpreter beim Beginn der Schleife ermittelt wurde. Die Ausgabe der Prozedur ist also

```
Error: Out of bounds with i=2
```

Lösung:
Für Anwendungen, in denen der Ende- oder Step-Ausdruck variabel ist, sollten Sie zu while- oder repeat-artigen Loop-Kontroll-Strukturen greifen.