10.5.5 GoTo

In den folgenden vier Kapiteln werden Kontroll-Strukturen beschrieben, deren Verwendung kontrovers diskutiert wird:

- GoTo — Unbedingter Sprung zu einem Ziel(-Label).
- GoSub — Aufruf eines lokalen Unterprogramms.
- On GoTo — Berechneter unbedingter Sprung zu einem ausgewählten Ziel(-Label).
- On GoSub — Berechneter Aufruf lokaler Unterprogramme.

Wie in den betrachteten Beispielen noch gezeigt werden kann, gibt es viele Anwendungsfälle, in denen die o.a. vier Anweisungen durch andere Kontroll-Strukturen ersetzt werden können. Oder anders formuliert: Die Verwendung der vier o.a. Anweisungen ist vermeidbar, wenn Sie die Komplexität des Gambas-Code nicht schreckt, um zum Beispiel eine dreifache For-Kontroll-Struktur bei einer bestimmten Bedingung geordnet zu verlassen.

10.5.5.1 GoTo-Syntax

Es wird zu einem Ziel(-Label) gewechselt, das an einer anderen Stelle in einer Prozedur oder Funktion deklariert ist:

```
GoTo Label
…
Label:
  Anweisung(en)
```

10.5.5.2 Hinweise zur Syntax

- Label zeigt hier auf ein Ziel für die GoTo-Anweisung. Die Ziel-Angabe endet mit einem Doppelpunkt. Sie können Ziel(-Label) durch die Schreibweise mit großen Buchstaben oder mit einem vorangestellten Unterstrich in besonderer Weise auffallend kennzeichnen.
- Ein Label ist stets lokal zu einer Funktion oder Prozedur.
- 'Label' im Zusammenhang mit der Anweisung 'GoTo Label' ist nicht zu verwechseln mit der Klasse Label → Kapitel 16.1 Label!
- GoTo und Label können verwendet werden, um eine Kontroll-Struktur zu verlassen.
- GoTo und Label können dagegen *nicht* verwendet werden, um in eine Kontroll-Struktur einzusteigen.

Das GoTo im Zusammenhang mit der o.a. Anweisung ist von der Methode *Editor.Goto(...)* der Komponente Editor (gb.qt4.ext) wohl zu unterscheiden:

```
Editor1.GoTo(0, String.Len(Editor1.Lines[0].Text)) ' Cursor an das Ende der 1.Zeile setzen
Editor1.GoTo(Editor1.Lines.Count, String.Len(Editor1.Text)) ' Cursor an das Textende setzen
```

Die Beispiele im nächsten Abschnitt zeigen vorwiegend Quelltext-Ausschnitte und werden hinreichend kommentiert.

10.5.5.3 Beispiel 1

Die Anweisung 'GoTo Label' ist in manchen Fällen als *ultima ratio* akzeptabel. Nämlich genau dann, wenn Sie zwei verschachtelte Schleifen verlassen möchten:

Variante 1 – ohne Verwendung der 'GoTo Label'-Anweisung

```
Dim i, j As Integer
Dim bBreak As Boolean

For i = 0 To 10
  For j = 0 To 10
    ' Die Doppel-Schleife soll verlassen werden, sobald i*j > 20 ist
    If (i * j > 20) Then
      bBreak = True
      Break
    Endif
  Next ' j
  If bBreak = True Then Break
Next ' i
```

- Sie können über 'Break' nur aus der innersten Schleife springen. Also müssen Sie eine Variable setzen, die der äußeren Schleife ebenfalls signalisiert, dass auch diese verlassen werden soll.
- Die Variable 'bBreak' ist nur dann True, wenn aus der inneren Schleife gesprungen wurde und damit auch die äußere Schleife verlassen wird. Ob der o.a Quelltext gut lesbar ist, müssen Sie für sich entscheiden.

Variante 2 – Einsatz der 'GoTo Label'-Anweisung

```
Dim i, j As Integer

For i = 0 To 10
  For j = 0 To 10
    If i * j > 20 Then GoTo LEAVEBOTH
  Next
Next

LEAVEBOTH:
Anweisung(en)

' Hier geht es weiter...
```

10.5.5.4 Beispiel 2

Ein Beispiel für eine Aufgabe, bei der die GoTo-Anweisung ihre Stärke demonstrieren kann, ist die Suche in einem N-dimensionalen Array nach einem bestimmten Integer-Wert – hier in einem zwei-dimensionalen Integer[][]-Array.

```
Dim iN, iI, iX As Integer

For iN = 0 To aArray.Max
  For iI = 0 To aArray[iN].Max
    If iX = aArray[iN][iI] Then GoTo _Found
  Next
Next

_Found:
' Hier kann man iN und iI verwenden. Gefunden wurde das Element, wenn iN <= aArray.Max
```

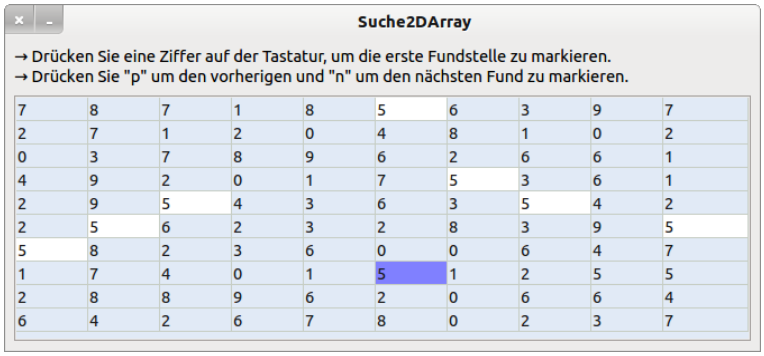

Abbildung 10.5.5.4.1: Suche nach einer (Integer-)Zahl in einem 2-dimensionalen Array

Das vollständige Projekt zum Beispiel 2 finden Sie im Download-Bereich.

Die einzige Möglichkeit im Beispiel 2 auf die GoTo-Anweisung zu verzichten, um bei Erfüllung einer bestimmten Bedingung aus zwei verschachtelten For-Kontroll-Strukturen zu springen, ist ein Flag:

```
Dim iN, iI As Integer
Dim bBreak As Boolean = False

For iN = 0 To aArray.Max
  For iI = 0 To Array[iN].Max
    If iX = aArray[iN][iI] Then
      bBreak = True
      Break
    Endif
  Next
  If bBreak Then Break
Next
```

Diese Lösung ist aber m.E. nicht gut, denn sie ist zum einen weniger gut lesbar und zum anderen weniger effizient. Für mehrere verschachtelte Schleifen wird es auf jeden Fall komplizierter. Ein ähnliches Problem tritt auf, wenn Sie beispielsweise über ein N-dimensionales Array iterieren wollen und N nur zur Laufzeit bekannt ist. Da eine For-Kontroll-Struktur nur linear aufzählen kann, benötigt man N verschachtelte For-Kontroll-Strukturen. Wenn N zur Laufzeit nicht bekannt ist, kann man ein dynamisch alloziertes Array von Iteratoren verwenden und mittels GoTo eine N- dimensionale *For-Kontrollstruktur* aufbauen. Dies wird z.B. bei der Generierung aller möglichen Zeichenketten mit N Buchstaben aus einem bestimmten Alphabet benötigt, wobei N vom Benutzer ausgewählt wird – so wie es beim 'Brute Forcing' praktiziert wird.