10.2.1 Bedingte Auswahl – IF-Kontroll-Struktur

In diesem Kapitel werden die folgenden Varianten der IF-Kontroll-Struktur beschrieben:

- Einseitige Auswahl IF…THEN…ENDIF
- Zweiseitige Auswahl IF…THEN…ELSE…ENDIF
- Mehrfach-Auswahl IF…THEN…[ELSE IF … .. ELSE IF] … ENDIF

10.2.1.1 Syntax

Die Syntax für die IF-Kontroll-Struktur im zweiten Teil steht für die Fälle, in denen jeweils nur eine (bedingte) Auswahl in einer Zeile formuliert wird:

```
IF Expression [ { AND IF | OR IF } Expression ... ] [ THEN ]
  ...
[ ELSE IF Expression [ { AND IF | OR IF } Expression ... ] [ THEN ]
  ... ]
[ ELSE
  ... ]
ENDIF
```

```
IF Expression [ { AND IF | OR IF } Expression ... ] THEN ...
IF Expression [ { AND IF | OR IF } Expression ... ] THEN ... ELSE …
```

10.2.1.2 Hinweise zur Syntax

- Für alle Operanden, die nicht vom Typ Boolean sind, benötigen Sie ein Relationszeichen, um einen Wahrheitswert von Ausdrücken (→ 10.2.1.6 Exkurs) zu erzeugen. Sie können diese Relationszeichen einsetzen: =, <>, <, >, <= und >= und nutzen die logischen Operatoren AND, OR und XOR sowie Klammern, um komplexe Ausdrücke zu notieren.
- Ein Ausdruck kann mit NOT *negiert* werden.
- Wenn kein Ausdruck wahr ist, dann wird die Else-Anweisung – wenn vorhanden – bedingungslos ausgeführt.
- Die IF-Kontroll-Struktur wird beendet, wenn nach dem ersten wahren Ausdruck die zugehörigen Anweisungen ausgeführt wurden oder nach der Ausführung der Anweisungen im bedingungslosen 'Else'-Zweig oder nach dem 'ENDIF'-Schlüsselwort.
- Die 'AND IF'- und 'OR IF'-Schlüsselworte dürfen nicht zusammen in einer Zeile stehen.
- Sie können IF-Kontroll-Strukturen auch verschachteln.
- Die Notation von *Then* nach *Expression* ist optional, wenn die nachfolgende Anweisung in einer weiteren Zeile steht.
- Die Schreibweisen 'Endif' und 'End If' sind gleichwertig.
- Seit Gambas 3.4 können Sie die IF-Kontroll-Struktur in einer Zeile notieren – vorausgesetzt, dass die nachfolgende Anweisung dem Schlüsselwort THEN folgt.

Die folgenden Beispiele zeigen vorwiegend Gambas-Quelltext-Ausschnitte und werden kurz kommentiert. Den Beispielen wird zum Teil Pseudo-Code vorangestellt, der als gut lesbare Interpretation der Syntax der IF-Kontroll-Struktur (→ 10.2.1.1 Syntax) gedacht ist.

10.2.1.3 Beispiele für einseitige Auswahl

Bei der einseitigen Auswahl wird die Anweisung oder der Anweisungsblock nur ausgeführt, wenn der Ausdruck wahr ist. Eine Alternative existiert nicht.

```
WENN Ausdruck wahr ist, DANN
     diese Anweisung(en) ausführen
ENDE der Auswahl
IF Len(txbFarbwert.Text) = 0 THEN
   txbFarbwert.MaxLength = 6
   Return
ENDIF
```

```
WENN Ausdruck wahr ist, DANN diese Anweisung(en) ausführen
IF Key.Code = Key.F1 THEN btnHelp_Click()
```

```
WENN Ausdruck NICHT wahr ist, DANN diese Anweisung(en) ausführen
IF NOT txbFarbwert.Text Then Message.Warning("Geben Sie einen Farbwert ein!")
```

10.2.1.4 Beispiele für zweiseitige Auswahl

Wenn Variante 1 über den RadioButton ausgewählt wurde, *dann* wird nach dieser Variante gerechnet, *sonst* nach der 2. Variante:

```
WENN Ausdruck wahr ist, DANN
     diese Anweisung(en) ausführen (und Kontroll-Struktur verlassen)
SONST
     jene Anweisung(en) ausführen
ENDE der Auswahl
IF optV1.Value = True THEN
   Calculate_V1(iAZahl, iEZahl)
ELSE
   Calculate_V2(iAZahl, iEZahl)
ENDIF
```

Die alternative Schreibweise der IF-Kontroll-Struktur in einer Zeile löst in der zweiten Zeile einen *Syntaxfehler* aus, weil in diesem Fall am Zeilen-Ende *kein* ENDIF stehen darf:

```
(1)    IF optV1.Value = True THEN Calc_V1(iAZahl, iEZahl) ELSE Calc_V2(iAZahl, iEZahl)
(2)    IF optV1.Value = True THEN Calc_V1(iAZahl, iEZahl) ELSE Calc_V2(iAZahl, iEZahl) ENDIF ' FEHLER!
```

10.2.1.5 Beispiele für Mehrfach-Auswahl

Hier eine IF-Kontroll-Struktur für drei (→ k=3) *unvereinbare* Fälle bei denen Sie sicher sind, dass genau einer der drei Fälle zutrifft:

```
WENN Ausdruck_1 wahr ist, DANN
     diese Anweisung(en)_1 ausführen (und Kontroll-Struktur verlassen)
SONST WENN Ausdruck_2 wahr ist, DANN
      diese Anweisung(en)_2 ausführen (und Kontroll-Struktur verlassen)
...
SONST WENN Ausdruck_k wahr ist, DANN
      jene Anweisung(en)_k ausführen
ENDE der Auswahl
Public Function Calculate(fP As Float, fQ As Float) As Variant[]
  Dim fDiskriminante As Float = 0
  Dim fX1, fX2 As Float
  Dim fXC1, fXC2 As Complex
  fDiskriminante = (fP * fP) / 4 - fQ
  IF Sgn(fDiskriminante) = 1 THEN ' D>0
     fX1 = - fP / 2 - Sqr(fDiskriminante)
     fX2 = - fP / 2 + Sqr(fDiskriminante)
     Return [fX1, fX2]
  ELSE IF Sgn(fDiskriminante) = 0 Then ' D=0
     fX1 = - fP / 2
     fX2 = fX1
     Return [fX1, fX2]
  ELSE IF Sgn(fDiskriminante) = -1 Then ' D<0
     fXC1 = Complex(- fP / 2, - Sqr(- fDiskriminante))
     fXC2 = fXC1.Conj()
     Return [fX1, fX2]
  ENDIF
End ' Calculate(fP As Float, fQ As Float) As Variant[]
```

Beispiel

```
WENN Ausdruck_1 wahr ist, DANN
     diese Anweisung(en)_1 ausführen (und Kontroll-Struktur verlassen)
SONST WENN Ausdruck_2 wahr ist, DANN
      diese Anweisung(en)_2 ausführen (und Kontroll-Struktur verlassen)
...
SONST
     bedingungslos jene Anweisung(en) ausführen!
ENDE der Auswahl
Public Sub GetStatus(sStatus As String)
  IF sStatus Like "[+Pp]*" THEN
     Message.Info(IIf(sStatus Begins "+", "Status: positiv", "Status: " & sStatus))
  ELSE IF sStatus Like "[-Nn]*" THEN
     If sStatus Begins "-" Then sStatus = "negativ"
     Message.Info("Status: " & sStatus)
  ELSE
     Message.Error("Der Status konnte NICHT ermittelt werden!")
  ENDIF
End ' GetStatus(sStatus As String)
```

Die bedingungslose Verwendung von ELSE wird oft für Fehler-Meldungen verwendet.

Beispiel

Mehrere über 'AND IF' oder 'OR IF' verknüpfte Ausdrücke werden in den nächsten Beispielen jeweils in einer Prozedur eingesetzt:

```
WENN Ausdruck_1 wahr ist UND WENN auch Ausdruck_2 wahr ist, DANN
     diese Anweisung(en) ausführen
ENDE der Auswahl

Public Sub txbFarbwert_KeyPress()
  IF (Key.Control AND Key.Code = Key.F1) THEN btnHelp_Click()

  IF (Key.Code = Key.Return OR Key.Code = Key.Enter) AND IF (txbFarbwert.Text) THEN
     Message.Info("Die eingegebenen Zeichen sind:\n\n" & Upper(txbFarbwert.Text))

     IF Left(txbFarbwert.Text, 1) = "&" THEN
        txbFarbwert.MaxLength = 7
     ELSE
        txbFarbwert.MaxLength = 6
     ENDIF ' Left(txbFarbwert.Text, 1) = "&" ?
  ENDIF ' Key.Code = Key.Return OR Key.Code = Key.Enter AND ...?
  IF (Key.Code = Key.BackSpace) AND IF (Len(txbFarbwert.Text) > 0) THEN
     txbFarbwert.Text = Left(txbFarbwert.Text, Len(txbFarbwert.Text) - 1)
  ENDIF ' Key.Code = Key.BackSpace AND Len(txbFarbwert.Text) > 0 ?

' Zulässige Zeichen für einen Farbwert in hexadezimaler Darstellung
  IF Key.Text NOT Like "[&0-9a-fA-F]" THEN
     Stop Event
  ENDIF ' Key.Text NOT Like "[&0-9a-fA-F]"

End ' txbFarbwert_KeyPress()
```

Beispiel

```
WENN Ausdruck_1 wahr ist ODER WENN Ausdruck_2 wahr ist, DANN
     diese Anweisung(en) ausführen
ENDE der Auswahl

Private Sub ResultSave()
  Dim sMessage1, sMessage2 As String

  Dialog.Path = sScriptFilePath
  Dialog.Title = ("Speichern Sie die Ergebnis-Matrix!")
  Dialog.Filter = ["*.fit", ("GnuPlot-Skript-Datei"), "*", ("Alle Dateien")]
  If Dialog.SaveFile() Then
     Message.Warning("Das Speichern wurde abgebrochen!") ' Abbrechen-Button wurde gedrückt!
     Return
  Else
     If (File.Ext(Dialog.Path)) OR IF (File.Ext(Dialog.Path) <> "fit") Then
        Dialog.Path = File.SetExt(Dialog.Path, "fit")
     Endif
     File.Save(Dialog.Path, txaErrorAndFit.Text)
     Finally
     btnSaveScriptFile.Enabled = False
     Catch
     sMessage1 = ("Die Ergebnis-Datei ")
     sMessage2 = (" kann NICHT gespeichert werden!")
     Message.Error("FEHLER!" & Chr(10) & sMessage1 & File.Name(Dialog.Path) & sMessage2)
  Endif ' Dialog.SaveFile() = True ?

End ' ResultSave()
```

Hinweise:

- OR und 'OR IF' als auch AND und 'AND IF' sind jeweils verschiedene Operatoren! OR und AND sind bitweise Operatoren, während 'OR IF' und 'AND IF' logische Operatoren sind.
- Sie können AND und OR beliebig in einem komplexen Ausdruck einsetzen, aber nur mit einer Variante der 'OR IF'- und 'AND IF'-Operatoren auf der gleichen Zeile!
- Wenn Sie in einem komplexen Ausdruck mehrere Ausdrücke mit dem 'AND IF'-Operator verknüpfen, dann werden diese von links nach rechts ausgewertet bis der erste Ausdruck FALSCH ist. Das bedeutet, dass der komplexe Ausdruck FALSCH ist. Nur wenn alle verknüpften Ausdrücke WAHR sind ist der komplexe Ausdruck WAHR.
- Wenn Sie in einem komplexen Ausdruck mehrere Ausdrücke mit dem 'OR IF'-Operator verknüpfen, dann werden diese von links nach rechts ausgewertet bis der erste Ausdruck WAHR ist. Das bedeutet, dass der komplexe Ausdruck WAHR ist. Nur wenn alle verknüpften Ausdrücke FALSCH sind ist auch der komplexe Ausdruck FALSCH.

10.2.1.6 Exkurs

Die folgenden Beispiele – aber auch einige der o.a. Beispiele – sind nur dann zu verstehen, wenn man den Inhalt des Begriffs *Expression* in Gambas genau kennt. In der Dokumentation zu 'Ausdruck' steht u.a.:

```
An expression is a value (a constant, a predefined constant, a variable or the result of a function), which
may optionally be preceded by certain operators depending on the type of value, followed by an operator and
another value, and so on.
```

Übersetzt und mit Beispielen versehen liest sich das so:

Ein Ausdruck ist ein Wert: Eine Konstante (*1), eine vordefinierte Konstante (*2), eine Variable (*3) oder das Ergebnis eines Funktionsaufrufs (*4), dem optional bestimmte Operatoren nach- oder voran-gestellt werden (*5) – je nach Typ des Wertes – und so weiter (*6).

Beispiele für die markierten Fälle (*1) bis (*6):

(*1) → 5 oder "Test"
(*2) → gb.Integer
(*3) → iIndex
(*4) → Pi(2)
(*5) → NOT bIsActive
(*6) → NOT bIsActive OR IF (iLevel < iMinLevel)

In Bezug auf *Ausdruck* im Zusammenhang mit der *IF-Kontroll-Struktur* gilt Folgendes:

Die IF-Kontroll-Struktur prüft den Wahrheitswert eines Ausdrucks. Schreibt man zum Beispiel als Ausdruck eine Integer-Variable, ist das nach der o.a. Definition immer noch ein Ausdruck und der Wahrheitswert eines Integers ist definiert als der Wahrheitswert der Relation *iInteger* <> *0*. Analog gilt für einen String: *sString* <> *Null*, wobei auch die leere Zeichenkette als Null gilt. Bei Objekten ist ebenfalls Null der charakteristische Wert für die Wahrheitswertbestimmung. In allen Fällen bedeutet 0 → False und alles andere ist True.

Beispiele

```
Public Sub btnTest_Click()
  Dim iInteger As Integer

  Print iInteger
  IF iInteger THEN
     Print "JA"
  ELSE
     Print "NEIN"
  ENDIF
'-------------------------------------------------------
  IF txbPingNumber.Text THEN
     Print "TEXTBOX ENTHÄLT TEXT."
  ELSE
     Print "TEXTBOX ENTHÄLT KEINEN TEXT."
  ENDIF
'-------------------------------------------------------
  IF txbPingNumber THEN
     Print "DAS OBJEKT 'txbPingCount' EXISTIERT"
  ELSE
     Print "DAS OBJEKT 'txbPingCount' EXISTIERT NICHT"
  ENDIF
'-------------------------------------------------------
  IF $hPing THEN
     Print "DER PING-PROZESS LEBT..."
  ELSE
     Print "VOM PING-PROZESS IST NICHTS ZU SEHEN..."
  ENDIF
'-------------------------------------------------------
  TRY File.Save("/usr/local/backup.bak", txaOutput.Text)
  IF ERROR THEN
     Print "Fehler: "; Error.Text; " - "; Error.Where
     Return
  ENDIF

End ' btnTest_Click()
```

ERROR allein ist ein Schlüsselwort der Sprache Gambas. Es wird True zurückgegeben, wenn die letzte TRY-Anweisung einen Fehler auslöste. Minisini rät dazu, ERROR nur in diesem Kontext zu verwenden.

Diese Ausgaben wurden in der Konsole der Gambas-IDE angezeigt:

```
0
NEIN
DIE TEXTBOX ENTHÄLT TEXT.
DAS OBJEKT 'txbPingCount' EXISTIERT
VOM PING-PROZESS IST NICHTS ZU SEHEN...
FEHLER: Access forbidden - FMain.btnTest_Click.135
```

- Die Variable wird mit 0 initialisiert und als Wert wird 0 ausgegeben.
- Das "NEIN" ist somit folgerichtig, weil mit *iInteger* <> *0* verglichen wird!
- In der GUI für das Konsolen-Programm wurde 4 als Vorgabewert für die Anzahl der Pings in die TextBox mit dem Namen *txbPingNumber* eingetragen. Dem trägt die Ausgabe Rechnung.
- Ein TextBox-Objekt mit dem Namen *txbPingNumber* existiert zur Laufzeit und wird so angezeigt.
- Ein Ping-Prozess mit '$hPing = Exec aCommand For Read As "myPingProcess"' wurde nicht gestartet – er existiert nicht. Die Ausgabe ist korrekt.
- Das Speichern der angegebenen Datei im vorgegebenen Pfad wurde *kontrolliert* angeschoben, schlug aber wegen fehlender Schreibrechte im Ordner fehl. Der Fehler wurde abgefangen und mit einer Fehlermeldung quittiert.