8.1 Zuweisungsoperatoren

In der Sprache Gambas ist der Zuweisungsoperator das Gleichheitszeichen =. Die Syntax für eine Zuweisungsoperation kann allgemein durch *Ziel = Ausdruck* beschrieben werden. Der Wert von Ausdruck kann einem der folgenden Ziele zugewiesen werden:

- lokale Variable,
- Funktionsparameter,
- globale Variable oder Klassen-Variable,
- Array,
- Object-Public-Variable,
- Objekteigenschaft.

Operator	Beschreibung
Ziel = Ausdruck	Direkte, einfache Zuweisung

Tabelle 8.1.1: Zuweisungs-Operator

Für die Zuweisungsoperation gilt die Rechtsassoziativität. Das bedeutet, das zuerst der Ausdruck auf der rechten Seite ausgewertet und dann dem Ziel auf der linken Seite zugewiesen wird.

Bitte beachten Sie, dass in den weiteren Abschnitten zum Kapitel 8 das Gleichheitszeichen bivalent auch als Vergleichsoperator eingesetzt wird.

8.1.1 Komplexe Zuweisungsoperatoren

Die komplexen oder zusammengesetzten Zuweisungsoperatoren repräsentieren nur verkürzte Schreibweisen von Zuweisungen mit speziellen Operatoren im Ausdruck:

Komplexer Zuweisungs-Operator	Beschreibung
Ziel += Expression	Synonym für die Zuweisung: Ziel = Ziel + Expression
Ziel -= Expression	Synonym für die Zuweisung: Ziel = Ziel - Expression
Ziel *= Expression	Synonym für die Zuweisung: Ziel = Ziel * Expression
Ziel /= Expression	Synonym für die Zuweisung: Ziel = Ziel / Expression
Ziel \= Expression	Synonym für die Zuweisung: Ziel = Ziel DIV Expression
Ziel %= Expression	Synonym für die Zuweisung: Ziel = Ziel MOD Expression
Ziel &= Expression	Synonym für die Zuweisung mit dem Operator & zum Verbinden von Zeichenketten: Ziel = Ziel & Expression
Ziel &/= Expression	Synonym für die Zuweisung mit dem Operator & zum Verbinden von Zeichenketten für Pfade: Ziel = Ziel &/ Expression

Tabelle 8.1.1.1: Komplexe Zuweisungsoperatoren

Beispiel:

```
[1]     Dim fNumber As Float
[2]
[3]     fNumber = 9.55
[4]     fNumber += 2 * (-3.21) ^ 3   ' Fehler: Print fNumber += 2*(-3.21)^3
[5]   ' fNumber = fNumber + 2 * (-3.21) ^ 3
[6]     Print fNumber
```

Kommentare:

- Das Gleichheitszeichen in der Zeile 3 ist der (einfache) Zuweisungs-Operator.
- In der 4. Zeile wird zu dem Wert von fNumber der Summand 2·(-3,21)^3 addiert und der Variablen fNumber als neuer Wert zugewiesen.
- Gleiches leistet die Zuweisung fNumber = fNumber + 2·(-3,21)^3
- Die Anweisung *PRINT fNumber += 2·(-3,21)^3* liefert einen Syntaxfehler.

Beispiele:

```
[1]   PUBLIC CONST DefaultStartX AS Float = -5.0
[2]   Private aMatrix As Variant[]
[3]   aMatrix = New Variant[]
[4]   Dim aNames As String[] = ["Hans", "Maria", "Peter", "Anna", "Robert"]
[5]   TableView1.Sorted = NOT TableView1.Sorted
[6]   TableView1.Columns.Count = 4
[7]   TableView1.Header = TableView1.Both
[8]   TableView1.Mode = Select.Single
[9]   Me.Flags[Editor.ShowLimits] = (iLimit > 0) AND (iLimit < 3)
[10]  btnTableViewExportCSV.Enabled = True
[11]  If Row MOD 2 = 0 Then TableView1.Data.Background = Color.RGB(224, 224, 224) ' hellgrau
```

Bitte beachten Sie: In der Zeile 11 ist das erste Gleichheitszeichen = ein Vergleichsoperator, während das 2. Gleichheitszeichen = ein Zuweisungsoperator ist. Die Hintergrundfarbe der Komponente TableView (Ziel) wird auf die Farbe hellgrau (Wert von Ausdruck) gesetzt.

8.1.2 Sonderfälle

Einige Anweisungen, die Werte zurück liefern wie SWAP, EXEC, SHELL, OPEN, NEW, RETURN oder RAISE verwenden oft ebenfalls die Zuweisungssyntax, wie die folgenden Beispiele belegen:

SWAP:

```
IF Printer.Portrait = TRUE THEN SWAP MargeB, MargeL
SWAP aIndexField[iRow].Field, aIndexField[iRow - 1].Field
SWAP iForeground, iBackground
```

SHELL und EXEC:

```
hWhichAsProcess = EXEC ["which", "gnuplot"] Wait For Read
hGnuPlot = SHELL "gnuplot" For Read Write As "hGnuplot"
```

OPEN:

```
hDatei = OPEN (User.Home &/ "V24T" &/ "v24T.conf") FOR CREATE
hFile = OPEN sDTDDateiPfad FOR CREATE ' DTD-Datei wird neu angelegt oder geleert
fFile = OPEN Dialog.Path FOR READ  ' Datendatei zum Lesen öffnen
TRY fFile = OPEN Dialog.Path FOR WRITE ' Datendatei zum Schreiben öffnen
```

NEW

```
$EditGrid = NEW TextBox(hParent) AS "EditGrid"
MyParentObserver = NEW Observer(Me.Parent) As "Parent"
IF NOT $cSlot.Exist(sSlot) THEN $cSlot[sSlot] = NEW Collection
hMenuItem = NEW Menu(hContext) As "mnuFileOpen"
```

RETURN – Rückgabe eines Funktionswertes

```
RETURN (Value - Me.MinValue) / (Me.MaxValue - Me.MinValue)
RETURN Potentiometer.Spin
IF hCtrl.Window.TopLevel THEN RETURN Object.Type(hCtrl.Window)
RETURN Sin(x) * Cos(x / 0.56) + (Pi / 3)
RETURN FALSE
```

RAISE

```
' Gambas class file
' © 2008 Daniel Fuchs

EXPORT
CREATE PRIVATE
INHERITS UserControl
' Die folgenden Ereignisse sind bereits in der Klasse UserControl enthalten:
EVENT Activate
EVENT MouseWheel() AS Boolean
```

```
...

PUBLIC SUB DrawingArea_MouseWheel()
  DIM Cancelled AS Boolean

  MySlideTimer.Stop
  Cancelled = RAISE MouseWheel()
  IF Cancelled OR NOT ME.Enabled THEN RETURN
  ME.Value = (ME.Value - Mouse.Delta * ME.GetSpin() * ME.PageStep)
  RAISE Activate

END ' DrawingArea_MouseWheel()
```