8.3 Operationen und Vergleiche für Zeichenketten

Für Operationen mit Zeichenketten stellt Gambas Ihnen nur 2 Operatoren zur Verfügung. Für die Vergleiche von zwei Zeichenketten werden Sie vor allen die Vergleichsoperatoren in der Tabelle 8.3.2.1 verwenden.

8.3.1 Operatoren für Zeichenketten

Gambas kennt die folgenden Operatoren für Operanden vom Typ Zeichenkette (String):

Operator	Beschreibung
String & String	Verbindet zwei Zeichenketten zu einer Zeichenkette
String &/ String	Verbindet zwei Zeichenketten, die Dateinamen enthalten. Es wird ein Pfad-Trennzeichen zwischen die beiden Zeichenketten eingefügt, wenn das erforderlich ist

Tabelle 8.3.1.1: Operatoren für Zeichenketten

```
Message.Error("Fehler: " & Chr(10) & "Zeichen nicht im Eingabe-Alphabet!")
If NOT Exist(User.Home &/ "V24T" &/ "v24T.conf") Then …
Path = SettingsP.DefaultDir &/ Application.Name & ".conf"
```

Beachten Sie, dass im dritten Beispiel das letzte & zwei Zeichenketten zu einem vollständigen Dateinamen (Datei-Name und Extension wie zum Beispiel: *udpserver.conf*) verbindet.

8.3.2 Vergleich von Zeichenketten

Von den Vergleichsoperatoren in der folgenden Tabelle sind vor allem die letzten drei interessant, weil sie in vielen Fällen durch die Angabe von Mustern effektiv eingesetzt werden können.

Vergleichsoperator	Beschreibung
String = String	Der Vergleich ist wahr, wenn zwei Zeichenketten exakt gleich sind
String == String	Der Vergleich ist wahr, wenn zwei Zeichenketten gleich sind, wobei beim Vergleich die Groß- und Kleinschreibung NICHT berücksichtigt wird
String <> String	Der Vergleich ist wahr, wenn zwei Zeichenketten unterschiedlich sind
String LIKE MusterString	Es wird geprüft, ob eine Zeichenkette mit einem Muster übereinstimmt
String BEGINS MusterString	Es wird geprüft, ob eine Zeichenkette mit dem Muster beginnt
String ENDS MusterString	Es wird geprüft, ob eine Zeichenkette mit dem Muster endet

Tabelle 8.3.2.1: Vergleichsoperatoren 1 für Zeichenketten

```
[1] Dim sZeichenkette As String
[2]
[3] sZeichenkette = "Gambas"
[4]
[5] Message.Info(IIF(sZeichenkette = "gambas", "Vergleich wahr.", "Vergleich falsch."))
[6] Message.Info(IIF(sZeichenkette == "gaMBas", "Vergleich wahr.", "Vergleich falsch."))
```

Kommentare:

- Das Gleichheitszeichen in der Zeile 1 ist ein Zuweisungs-Operator.
- Im ersten Vergleich *IF sZeichenkette = "gambas"* in der 5. Zeile wird das Gleichheitszeichen als Vergleichsoperator eingesetzt. Der Vergleich liefert eine falsche Aussage.
- In der 6. Zeile liefert der Vergleich *IF sZeichenkette == "gaMBas"* eine wahre Aussage, weil beim Vergleich die Groß- und Kleinschreibung NICHT berücksichtigt wird.

Während die Vergleichsoperatoren = und <> häufig eingesetzt werden, finden die folgenden Vergleichsoperatoren relativ selten Verwendung.

Vergleichsoperator	Beschreibung
String1 < String2	Der Vergleich ist wahr, wenn Zeichenkette1 kleiner als Zeichenkette2 ist
String1 > String2	Der Vergleich ist wahr, wenn Zeichenkette1 größer als Zeichenkette2 ist
String1 <= String2	Der Vergleich ist wahr, wenn Zeichenkette1 kleiner oder gleich als Zeichenkette2 ist
String1 >= String2	Der Vergleich ist wahr, wenn Zeichenkette1 größer oder gleich als Zeichenkette2 ist

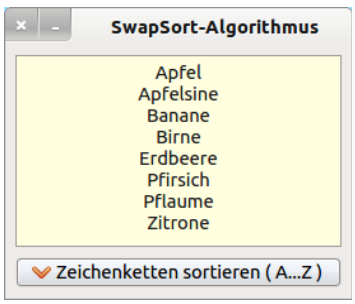
Tabelle 8.3.2.2: Vergleichsoperatoren für Zeichenketten

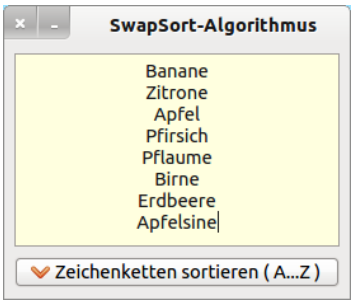
Die Vergleichsoperatoren <= und >= gibt es zum Beispiel in der Klasse *app/src/gambas3/.src/Component/CSymbolInfo.class* und wird dort benutzt um festzustellen, ob ein einzelnes Zeichen, das in dem String *sCar* gespeichert ist, zu einem bestimmten Intervall A bis Z im ASCII-Zeichensatz gehört.

```
If (sCar >= "A" AND sCar <= "Z") OR sCar = "." Then ...
```

Wenn Sie zum Beispiel Zeichenketten mit eigenen Prozeduren sortieren wollen, führt das über die o.a. Vergleichsoperatoren für Zeichenketten, wie Sie dem folgenden Quelltext entnehmen können:

```
Public Sub Form_Open()
  FMain.Center
  FMain.Resizable = False
End ' Form_Open()

Public Sub btnSort_Click()
  Dim aStringArray As String[]

  aStringArray = Split(txaText.Text, "\n")
    SwapSortStrings(aStringArray)
  txaText.Text = aStringArray.Join("\n")

End ' btnSort_Click()

Private Sub SwapSortStrings(aMatrix As String[])
' Implementation SwapSort-Algorithmus nach http://de.wikipedia.org/wiki/Swap-Sort
  Dim i, j, m As Integer
  Dim sTemp As String

  i = 0
  While i < aMatrix.Count
    m = 0
    For j = 0 To aMatrix.Count - 1
      If aMatrix[j] < aMatrix[i] Then Inc m ' Vergleich von Zeichenketten
    Next ' j
    If i = m Then
       Inc i
    Else
       sTemp = aMatrix[m]
       aMatrix[m] = aMatrix[i]
       aMatrix[i] = sTemp
    Endif ' i = m?
  Wend
End ' SwapSortStrings(...)
```





Abbildungen 8.3.2.1 und 8.3.2.2: Sortierung von Zeichenketten

Ein Nachteil von *SwapSort:* Jedes Element in der zu sortierenden Liste darf nur einmal vorkommen. Die Methode *String[].Sort()* besitzt diesen Nachteil nicht und benutzt den QuickSort-Algorithmus.

8.3.3 Operator LIKE

```
bErgebnis = String [NOT] LIKE Muster
```

Die boolsche Variable *bErgebnis* erhält den Wert TRUE, wenn die Muster-Zeichenkette auf *String* passt. Wird NOT verwendet, dann wird der Test invertiert.

Der Operator ist nicht case-sensitive und es können die folgenden Muster verwendet werden:

Muster	Interpretation
*	Eine Anzahl von beliebigen Zeichen
?	Ein einzelnes Zeichen
[abc]	Ein Zeichen aus der Menge in den eckigen Klammern
[x-y]	Ein Zeichen aus dem Intervall in den eckigen Klammern
[^x-y]	Ein Zeichen nicht aus dem Intervall in den eckigen Klammern
space	Jede Anzahl von Zeichen mit einem ASCII-Code kleiner als 32
{aaa,bbb,...}	Eine der Zeichenketten aus der { Menge }, die durch Komma getrennt werden.
\x	Spezielle Zeichen x haben für den Operator LIKE besondere Bedeutung und man nutzt \x, um solche speziellen Zeichen darzustellen.

Tabelle 8.3.3.1: Muster

Beispiel	Wahrheitswert	Beschreibung
"Gambas" LIKE "G*"	TRUE	Der String Gambas beginnt mit G, dem beliebig viele Zeichen folgen (können).
"Gambas" LIKE "G[^Aa]*"	FALSE	Gambas beginnt mit G, dem kein A oder a folgen darf, was zum Fehler führt. Das * wird nicht weiter beachtet.
"Gambas" LIKE "?[Aa]*"	TRUE	Nach *einem* beliebigen Zeichen, dem ein a oder A folgen muss, stehen beliebig viele andere Zeichen.
sDevice NOT LIKE "dev/ttyUS{B0,B1}"	TRUE	Annahme: Die Variable sDevice hat den Wert "dev/ttyUSB2". Kein Element aus der Muster-Menge passt.

Tabelle 8.3.3.2: Vergleichsoperatoren 1 für Zeichenketten

Bitte beachten Sie: LIKE gilt nur für ASCII-Strings. Wenn Sie gegen UTF-8-Strings testen, müssen Sie die *gb.pcre* Komponente verwenden.

8.3.4 Operator BEGINS

```
bResult = String [NOT] BEGINS Muster
```

Die boolsche Variable *bResult* erhält den Wert TRUE, wenn *String* mit der Muster-Zeichenkette beginnt. Wird NOT verwendet, dann wird der Test invertiert. Der Operator BEGINS ist case-sensitive.

Beispiel	Wahrheitswert	Beschreibung
"Gambas" BEGINS "Gam"	TRUE	Die Zeichenkette *Gambas* beginnt mit der Zeichenkette *Gam.*
"Gambas" BEGINS "Ham"	FALSE	*Gambas* beginnt mit nicht mit der Zeichenkette *Ham.*
"Hans" NOT BEGINS "Ga"	TRUE	Stimmt; *Hans* beginnt NICHT mit der Zeichenkette *Ga.*

Tabelle 8.3.4.1: Beispiele für den Operator BEGINS

8.3.5 Operator ENDS

```
bResult = String [NOT] ENDS Muster
```

Die boolsche Variable *bResult* erhält den Wert TRUE, wenn *String* mit der Muster-Zeichenkette endet. Wird NOT verwendet, dann wird der Test invertiert. Der Operator ENDS ist case-sensitive.

Beispiel	Wahrheitswert	Beschreibung
"Gambas" ENDS "bas"	TRUE	Die Zeichenkette *Gambas* endet auf bas.
"Gambas" ENDS "BAS"	FALSE	Die Zeichenkette *Gambas* endet NICHT auf BAS.
"Hans" NOT ENDS "ins"	TRUE	Stimmt; *Hans* endet NICHT auf *ins.*

Tabelle 8.3.5.1: Beispiele für den Operator ENDS