19.6.3 Exkurs LIKE

Das Thema "Operator LIKE" aus dem Kapitel 8.8 wird hier unter dem *Anwendungsaspekt* wieder aufgegriffen. Es gilt:

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

und die boolsche Variable *bErgebnis* erhält den Wert TRUE, wenn die Muster-Zeichenkette auf *String* passt. Wird NOT verwendet, dann wird der Test invertiert. Es können die folgenden Muster verwendet werden:

Muster	Beschreibung: Passt genau auf … oder ist Platzhalter für genau ...
*	eine Anzahl von beliebigen Zeichen
?	ein einzelnes Zeichen
[abc]	ein Zeichen aus der Zeichen-Menge
[x-y]	ein Zeichen aus dem vorgegebenen Bereich
[^x-y]	jedes Zeichen, das *nicht* im Bereich ist; ^ ist hier das Negationszeichen – nicht !
space	jede Anzahl von Zeichen mit einem ASCII-Code kleiner als 32
{aaa,bbb,...}	eines der 'Worte' in der durch Komma getrennten Liste

Tabelle 19.6.3.1: Muster für LIKE

Beispiel 1
In einem Datei-Öffnen-Dialog kann man ein eigenes Datei-Filter einsetzen. Dabei wird für Bilddateien oft ein Filter der folgenden Art verwendet:

```
Dialog.Filter = ["*.png;*.jpg", "Picture files", "*.svg;*.wmf", "Drawing files"]
```

Mit regulären Ausdrücken können Sie jedoch die Filter wesentlich differenzierter setzen:

```
Dialog.Title = "Importieren Sie eine Tabellen-Text-Datei!"
Dialog.Filter = ["tb*.txt", "Text-Dateien"] ' Datei beginnt mit tb
Dialog.Filter = ["[0-9]*.txt", "Text-Dateien"] ' Datei beginnt mit einer Ziffer
Dialog.Filter = ["tb[1-3]*.txt; ta[1-3]*.log", "Text-Dateien"]
```

Mit dem letzten Filter werden alle Dateien im aktuellen Verzeichnis selektiert, die mit der Buchstabenfolge 'tb' beginnen, der eine Ziffer aus dem Bereich von 1-3 folgt und dann eine Folge von beliebigen Zeichen. Die Extension ist entweder .txt oder .log.

Die folgende Filter-Definition ergibt mit Gambas 3.4.0 einen Fehler (FileView.CheckFilter.127: Bad regular expression:), weil nach Definition immer nur ein Filter gesetzt werden kann:

```
Dialog.Filter = ["tb[1-3]*.{txt,log}", "Text-Dateien"]
```

Beispiel 2
Seitennamen filtern Sie zum Beispiel so:

```
FOR EACH sLine IN Split(sCont, "\n")
    IF sLine LIKE "/wiki/" & CGI.Encode(Name2Wiki(TextBox2.Text)) & "*" THEN
       aUrls.Add(sLine)
    ENDIF
NEXT ' sLine
```

Für ein starkes Passwort könnte die Forderung nach jeweils mindestens einer Ziffer, einem kleinen Buchstaben, einem Großbuchstaben und einem ausgewählten Sonderzeichen mit dieser Funktion geprüft werden und der Funktionswert für eine *differenzierte Fehleranalyse* genutzt werden:

```
Private Function CheckStrongPassword(sPassword As String) As Integer

  If sPassword NOT LIKE "*[a-z]*" Then Return 1
  If sPassword NOT LIKE "*[A-Z]*" Then Return 2
```

```
  If sPassword NOT LIKE "*[0-9]*" Then Return 3
  If sPassword NOT LIKE "*[+#%]*" Then Return 4 ' Zeichenmenge {+ # %}
  If Len(sPassword) < 8 Then Return 5

  Return 0 ' Das Password ist ein starkes Password

End ' Function CheckStrongPassword(..)
```

Beispiel 3
Unter der Annahme, dass die Variable *sDevice* den aktuellen Wert 'dev/ttyUSB2' hat, passt kein Element aus der Muster-Menge.

```
If sDevice LIKE "dev/ttyUS{B0,B1}" Then ...
```