8.4 Logische Operatoren

Die Gruppe der logischen Operatoren NOT, AND, OR und XOR wird einerseits für Ausdrücke vom Typ Wahrheitswert (boolean) benötigt und andererseits für Zahlen vom Typ Short, Integer oder Long(-Integer). Die logischen Operatoren bilden die Grundlage für effiziente Vergleiche und für die Funktionen zur Bit-Manipulation. Spezialfälle betreffen Zeichenketten und Objekte bei der logischen Negation mit dem Operator NOT. Das Setzen von Klammern beeinflusst die Rangfolge der Operationen, wenn mehrere logische Operatoren verwendet werden.

Operator	Beschreibung
NOT Ausdruck0	Berechnet das logische NOT des Ausdrucks (aus True wird False und umgekehrt) oder einer Zahl auf der Grundlage der binären Darstellung dieser Zahl.
Ausdruck1 AND Ausdruck2	Berechnet das logische AND zweier Ausdrücke vom Typ Boolean oder das numerische AND zweier Integer-Zahlen auf der Grundlage der binären Darstellung dieser Zahlen.
Ausdruck1 OR Ausdruck2	Berechnet das logische OR zweier Ausdrücke vom Typ Boolean oder das numerische OR zweier Integer-Zahlen auf der Grundlage der binären Darstellung dieser Zahlen.
Ausdruck1 XOR Ausdruck2	Berechnet das logische XOR zweier Ausdrücke vom Typ Boolean oder das numerische XOR zweier Integer-Zahlen auf der Grundlage der binären Darstellung dieser Zahlen.

Tabelle 8.4.1: Logische Operatoren für Ausdrücke vom Typ Boolean oder Zahlen

Beispiele:

```
If (sCar >= "A" AND sCar <= "Z") OR sCar = "." Then ...
If sPort AND sPort <> "80" Then sReq &= ":" & sPort
IF (x < 0) OR (x <> Round(x)) Then Error.Raise("Mathematic error")
Do While (sSocket.Status <> Net.Connected AND sSocket.Status > 0)
```

8.4.1 Logische Operatoren für Zahlen

Die Zahlen sind vom Typ Boolean Short, Integer oder Long(-Integer). Die Berechnung mit dem verwendeten logischen Operator erfolgt auf der *bitweisen* logischen Operation der binären Darstellung der Zahlen auf der Grundlage der sogenannten Wahrheitstafeln.

Der Operator invertiert jedes Bit in der binären Darstellung der Zahl, wie es in der folgenden Tabelle und in den Beispielen dargestellt wird:

Bit 1	Bit 2	Not Bit 1	Bit1 AND Bit2	Bit1 OR Bit2	Bit1 XOR Bit2
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Tabelle 8.4.1.1: Logische Operatoren und Wahrheitstafel

Beispiele:

```
Print 13;; Bin(13, 8);; Bin(Not 13, 8);; NOT 13
      13   00001101     11110010         -14
Print 5;; 12;; Bin(5, 8);; Bin(12, 8);; Bin(5 AND 12, 8);; 5 AND 12
      5    12   00000101    00001100     00000100           4
Print 5;; 12;; Bin(5, 8);; Bin(12, 8);; Bin(5 OR 12, 8);; 5 OR 12
      5    12   00000101    00001100     00001101          13
Print 5;; 12;; Bin(5, 8);; Bin(12, 8);; Bin(5 XOR 12, 8);; 5 XOR 12
      5    12   00000101    00001100     00001001           9
```

Im Kapitel *9.9 Bit-Manipulation* werden Funktionen vorgestellt, die konsequent mit den vorgestellten logischen Operatoren arbeiten.

8.4.2 Spezialfall Objekt und Zeichenkette

Wenn der Operand nach dem unären logischen Operator NOT eine *Zeichenkette* (string) oder ein *Objekt* ist, dann gilt folgendes für *NOT Ausdruck*: Ist der Ausdruck eine Zeichenkette oder ein Objekt, wird *True* zurückgegeben, wenn der Ausdruck Null ist oder *False* wenn der Ausdruck NICHT Null ist.

Beispiele:

```
Sub Picture_Write(pPicture As Picture)
    If NOT pPicture Then pPicture = HorizontalFader.DefaultPicture
    MyOriginalPicture = pPicture
    If MyOriginalPicture Then GetPictures
    Me.Draw
End ' Picture_Write(pBild As Picture)
```

```
If NOT btnOhneText.Text Then Message.Info("Button OHNE Beschriftung!")
```

- Wenn kein Bild übergeben wurde, dann wird ein Standard-Bild zugewiesen.
- Da der Button im letzten Beispiel offensichtlich nicht beschriftet wurde, hat der negierte Ausdruck *NOT btnOhnetext.Text* den Wahrheitswert True, weil der Ausdruck garantiert Null ist.