

## Logik

**Logik, Wahrheitstafeln, logische Terme und Regeln ..** Haftendorn 2011

Logische Terme braucht man vor allem bei Verzweigungen und Schleifen.  
 if Bedingungen oder While Bedingung  
 Der Gebrauch ist in der Lerndatei programmieren-einf-tns beschrieben.  
 Hier geht es also um die **Bedingungen**

**if ant="ja" then** könnte da z.B. stehen. Vorher muss dann eine Variable mit der Zeichenkette "ja" belegt sein.

Typisch ist auch **while eps>0.001 Block endwhile** (in 3 Zeilen geschrieben)  
 Dann wird der Anweisungsblock solange ausgeführt, wie die Bedingung erfüllt ist-  
 Oft braucht man aber eine Doppelbedingung der Form  
**while eps>0.001 or n <20 Block endwhile** (in 3 Zeilen geschrieben),  
 also einen logischen Term mit **or**. Damit wird abgesichert, dass der Schleifendurchlauf abgebrochen wird, obwohl eps aus irgendeinem Grund (z.B. Programmierfehler) nicht klein wird. Dazu muss dann aber die Variable n wirklich bei jedem Durchlauf um 1 größer werden. Die übernächsten nachfolgenden Seiten gehen auf die **Logik** als mathematisches Teilgebiet ein.

1.1

**Einige Beispiele für den Logik Gebrauch**

$$f(x) := \begin{cases} x^2 \cdot x < -2 \text{ or } x > 3 \\ 1, -2 \leq x \leq 3 \end{cases} \cdot \text{Fertig}$$

$$f(-3) \cdot 9 \cdot f(2) \cdot 1 \cdot f(-2.0001) \cdot 4.0004 \cdot f(3) \cdot 1 \cdot f(3.0001) \cdot 9.0006$$

Diese Funktion ist also außen eine Parabel und im Intervall [-2,3] eine waagerechte Gerade. (Übrigens diese Form für f ist bei den Vorlagen!!!!)

Anstelle von **or** darf oben natürlich nicht **and** stehen. Es gibt kein x, das sowohl links als auch rechts auf dem Zahlenstrahl liegt. Umgangssprachlich sagt man evtl. "Die Parabel gilt für  $x < -2$  **und**  $x > 3$ ". Darum muss man sich mit Logik befassen. Die nachfolgenden Seiten gehen auf die **Logik** als mathematisches Teilgebiet ein.

Gleichungen sind übrigens **logische Aussagen** wenn sie WAHR oder FALSCH sein können.  $3=4 \cdot \text{false}$   $3=3 \cdot \text{true}$ . Gleichungen die Variablen einhalten heißen **Aussageformen**  $x>3 \cdot x>3$  Sie werden erst beim Einsetzen Aussagen.

1.2

**Logik, Wahrheitstafeln, logische Terme und Regeln ..** Haftendorn 2011

Man kann einzel prüfen true or false  $\rightarrow$  true Gleichbedeutend mit 1 or 0  $\cdot$  1  
 Hier sind die üblichen Wahrheitstafeln nachgebildet.

Define **wtt(a,b,c)=Func** • Fertig

```

Local ma
ma:=newMat(3,3)
ma[1,1]:=a; ma[1,2]:=b; ma[1,3]:=c
ma[2,1]:=a and b; ma[2,2]:=a and c; ma[2,3]:=b and c
ma[3,1]:=a or b; ma[3,2]:=a or c; ma[3,3]:=b or c
Return ma
EndFunc
    
```

**wtt(true,false,false)**  $\cdot$   $\begin{bmatrix} \text{true} & \text{false} & \text{false} \\ \text{false} & \text{false} & \text{false} \\ \text{true} & \text{true} & \text{false} \end{bmatrix}$   $\begin{bmatrix} a & b & c \\ a \text{ and } b & a \text{ and } c & b \text{ and } c \\ a \text{ or } b & a \text{ or } c & b \text{ or } c \end{bmatrix}$

**wtt(true,true,false)**  $\cdot$   $\begin{bmatrix} \text{true} & \text{true} & \text{false} \\ \text{true} & \text{true} & \text{false} \\ \text{true} & \text{true} & \text{true} \end{bmatrix}$

1.3

**Logische Terme ..**

**wt(a,b,c)** • Fertig

```

Local ma
ma:=newMat(3,3)
ma[1,1]:=a; ma[1,2]:=b; ma[1,3]:=c
ma[2,1]:=a and b; ma[2,2]:=a and c; ma[2,3]:=a and b and c
ma[3,1]:=a or b; ma[3,2]:=a and (b or c); ma[3,3]:=a and b or a and c
Return ma
EndFunc
    
```

**wt(true,false,true)**  $\cdot$   $\begin{bmatrix} \text{true} & \text{false} & \text{true} \\ \text{false} & \text{true} & \text{true} \\ \text{true} & \text{true} & \text{true} \end{bmatrix}$

Man kann hier und auf der folgenden Seite die Wahrheitswerte der Terme ablesen.

1.4

**wt(true,true,true)**  $\cdot$   $\begin{bmatrix} \text{true} & \text{true} & \text{true} \\ \text{true} & \text{true} & \text{true} \\ \text{true} & \text{true} & \text{true} \end{bmatrix}$  **wt(true,true,false)**  $\cdot$   $\begin{bmatrix} \text{true} & \text{true} & \text{false} \\ \text{true} & \text{true} & \text{true} \\ \text{true} & \text{true} & \text{true} \end{bmatrix}$

**wt(true,false,true)**  $\cdot$   $\begin{bmatrix} \text{true} & \text{false} & \text{true} \\ \text{false} & \text{true} & \text{true} \\ \text{true} & \text{true} & \text{true} \end{bmatrix}$  **wt(false,true,true)**  $\cdot$   $\begin{bmatrix} \text{false} & \text{true} & \text{true} \\ \text{false} & \text{true} & \text{true} \\ \text{true} & \text{false} & \text{false} \end{bmatrix}$

**wt(false,true,false)**  $\cdot$   $\begin{bmatrix} \text{false} & \text{true} & \text{false} \\ \text{false} & \text{false} & \text{false} \\ \text{true} & \text{false} & \text{false} \end{bmatrix}$  **wt(false,false,false)**  $\cdot$   $\begin{bmatrix} \text{false} & \text{false} & \text{false} \\ \text{false} & \text{false} & \text{false} \\ \text{false} & \text{false} & \text{false} \end{bmatrix}$

**wt(false,false,true)**  $\cdot$   $\begin{bmatrix} \text{false} & \text{false} & \text{true} \\ \text{false} & \text{false} & \text{false} \\ \text{false} & \text{false} & \text{false} \end{bmatrix}$  **wt(true,false,false)**  $\cdot$   $\begin{bmatrix} \text{true} & \text{false} & \text{false} \\ \text{false} & \text{true} & \text{true} \\ \text{true} & \text{false} & \text{false} \end{bmatrix}$

**wt(false,false,false)**  $\cdot$   $\begin{bmatrix} \text{false} & \text{false} & \text{false} \\ \text{false} & \text{false} & \text{false} \\ \text{false} & \text{false} & \text{false} \end{bmatrix}$

1.5

Hier ist noch **xor** mit eingebaut

**wie(a,b,c)** • Fertig

```

Local ma
ma:=newMat(4,3)
ma[1,1]:=a; ma[1,2]:=b; ma[1,3]:=c
ma[2,1]:=a and b; ma[2,2]:=a and c; ma[2,3]:=a and b and c
ma[3,1]:=a or b; ma[3,2]:=a and (b or c); ma[3,3]:=a and b or a and c
ma[4,1]:=a xor b; ma[4,2]:=a and (b xor c)
Return ma
    
```

**wie(3-4,3-3,3-9)**  $\cdot$   $\begin{bmatrix} \text{false} & \text{true} & \text{false} \\ \text{false} & \text{false} & \text{false} \\ \text{true} & \text{false} & \text{false} \\ \text{true} & \text{false} & \text{false} \end{bmatrix}$  **wie(3-4,3-3,3-3)**  $\cdot$   $\begin{bmatrix} \text{false} & \text{true} & \text{true} \\ \text{true} & \text{false} & \text{true} \\ \text{true} & \text{false} & \text{false} \\ \text{true} & \text{false} & \text{false} \end{bmatrix}$

1.6

**De Morgansche Regel ..**

nicht (a und b) vgl nicht a und nicht b vgl nicht a oder nicht b vgl nicht (a oder b)  
 © Also gilt [not (a and b)]=not a or not b und [not (a or b)]=not a and not b ] Gesetze von **de Morgan**

**a,b,not (a and b)=not a or not b, not (a or b)=not a and not b**

**morgan(true,false)**  $\cdot$  { true,false,true, "=",true,false, "=",false }

**morgan(true,true)**  $\cdot$  { true,true,false, "=",false,false, "=",false }

**morgan(false,false)**  $\cdot$  { false,false,true, "=",true,true, "=",true }

© **Implikation A,B,  $\neg A \vee B$  aus A folgt B**

**folgt(false,true)**  $\cdot$  { false,true,true } **folgt(true,false)**  $\cdot$  { true,false,false }

**folgt(false,false)**  $\cdot$  { false,false,true } **folgt(true,true)**  $\cdot$  { true,true,true }

Äquivalenz **A,B,  $A \& B \vee \neg A \& \neg B$**

**aeq(true,false)**  $\cdot$  { true,false,false } **aeq(true,true)**  $\cdot$  { true,true,true }

**aeq(false,false)**  $\cdot$  { false,false,true } **aeq(false,true)**  $\cdot$  { false,true,false }

1.7

**Define LibPub morgan(a,b)=Func** • Fertig

© Aufruf **morgan(true,false)** zum Beispiel

Disp "a,b,not (a and b)=not a or not b, not (a or b)=not a and not b "

{ a,b,not (a and b),"=",not a or not b,not (a or b),"=",not a and not b }

EndFunc

**Define LibPub folgt(a,b)=Func** • Fertig

Disp "A,B,  $\neg A \vee B$ "

{ a,b,not a or b }

EndFunc

**Define LibPub aeq(a,b)=Func** • Fertig

Disp "A&B  $\vee \neg A \& \neg B$ "

{ a,b,a and b or not a and not b }

EndFunc

1.8