Übungsblock 13

Ein- und Ausgaben in Prolog:

```
?- write(2+3).
2+3
?- write(hugo).
hugo
?- write(Hugo).
_G180
Hugo = _G180
?- write(‘Hugo‘).
Hugo
?- X is 5, write(X).
5
X = 5
?- write(‘Zeilenumbruch‘),nl,write(‘in Prolog‘).
Zeilenumbruch
in Prolog
```

```
mult(X,Y):-
    Ergebnis is X*Y,
    write(X*Y),write(‘=‘),write(Ergebnis)
```

Übungsblock 13

Ein- und Ausgaben in Prolog:

```
?-read(X).
|: peter.

peter
X = peter
```

Term wird über Tastatur gelesen, muss mit Punkt und Zeilenumbruch oder Leerzeichen abgeschlossen werden!
Achtung: Anfangsbuchstabe groß → Variable
Achtung: syntaktisch muss Eingabe gültiger Prolog-Term sein

Übungsblock 13

Arbeiten mit Dateien:

```
druckeDatei(D) :- open(D,read,Stream),
                  repeat, read(Stream,T), write(T),nl,
                  at_end_of_file(Stream),
                  close(Stream).
```

open(Dateiname,mode,Stream): Datei namens D mit Modus mode (read, write, append, update) geöffnet, Stream erzeugt (in Stream-Var.),
close(Stream): aktueller Stream wird geschlossen,

at_end_of_file(Stream): beweisbar, wenn Dateiende erreicht

read, write: Lesen, Schreiben von Standard-In/Out bzw. von angegebenem Stream
ABER: read liest nur Prolog-Terme!

Übungsblock 13

Ein- und Ausgaben in Prolog:

```
vater :- write('Von welcher Person möchten Sie den Vater?'),nl,
         read(Person), vater(V,Person),
         write('Der Vater von '),write(Person), write(' ist '),
         write(V),write('. '),nl.
```

Weitere:

get (N)	liest ein Zeichen über Tastatur, unifiziert ASCII-Code des Zeichens mit N, Zeichen muss druckbar (ASCII > 32) sein
get0(N)	wie get(N), aber auch nicht druckbare Zeichen werden gelesen, analog: get_code(N)
get_char(N)	wie get0(N), N wird mit Zeichen unifiziert
skip(N)	liest aus Eingabe, bis ASCII-Wert des Zeichens gleich N ist
put(N)	gibt Zeichen aus, dessen ASCII-Wert N ist
put_code(N)	analog
put_char(N)	gibt Zeichen aus, das mit Variable N unifiziert ist

Übungsblock 13

Weitere:

tab(N) gibt N Leerstellen aus

display(X) gibt Argument in Präfix-Schreibweise aus

```
?-display(3+4*5).
+(3, *(4,5))
Yes
```

name(X,Y) gibt zum Atom X seine Liste von ASCii-Nummern aus (oder umgekehrt).

```
?- name(hugo,L)
L = [104, 117,103,111]
```

writeq(X) wie write, gibt Argument in Apostrophs aus wo dies notwendig ist

```
?- writeq(‘prolog‘).
prolog
?- writeq(‘Prolog‘).
‘Prolog‘
```