Dateien und Module

Heute:

- Ein- und Ausgabe
- Module

Ausgabe auf den Bildschirm

```
?- X=’Hagrid’, write(’Hallo ’), write(X).
Hallo Hagrid

X = ’Hagrid’
Yes
?- write(ron),nl,write(hermione).
ron
hermione

Yes
?- write(ron),tab(5),write(hermione).
ron     hermione

Yes
```

Ausgabe in Dateien

Schritt 1: Zur Datei wird ein Stream geöffnet.

Schritt 2: Schreiben auf diesen Stream.

Schritt 3: Stream wird wieder geschlossen.

Ausgabe in Dateien: Schritt 1

`open(` *+Dateiname*, *+Modus*, *-StreamID* `)`

Modus ist `read`, `write`, oder `append`.

```
?- open('the monster book of monsters', write, Stream).
Stream = '$stream'(191536)
Yes
```

Ausgabe in Dateien: Schritt 2

Eingebaute Prädikate: `write/2, tab/2, nl/1`.

Machen das Gleiche wie `write/1, tab/1, nl/0`, wobei das erste Argument jeweils der Identifikator des Streams ist, auf den geschrieben werden soll.

```
?- write('$stream'(191536), 'flobberworms').
Yes
3 ?- tab('$stream'(191536), 5),
     write('$stream'(191536), 'blast-ended skrewts').
Yes
4 ?- nl('$stream'(191536)),
     write('$stream'(191536), unicorns).
Yes
```

Ausgabe in Dateien: Schritt 3

close(+*StreamID*)

```
?- close('$stream'(191536)).
Yes
?- write('$stream'(191536), hippogriffs).
ERROR: stream '$stream'(191536) does not exist
```

Die Datei 'the monster books of monsters':

```
flobberworms     blast-ended skrewts
unicorns
```

Ausgabe in Dateien: alles

```
?- open('the monster book of monsters', write, Stream),
   write(Stream, 'blah blah'),
   close(Stream).
```

Eingabe vom Bildschirm lesen (1)

`read(`*?Term*`)`: Fordert den Benutzer auf eine Eingabe zu machen und liest dann einen Prologterm.

```
?- read(T).
|: hagrid.
T = hagrid
Yes
?- read(T).
|: giant(hagrid).
T = giant(hagrid)
Yes
?- read(T).
|: [a,b,c].
T = [a, b, c]
Yes
```

Eingabe vom Bildschirm lesen (2)

`get0(`*?Char*`)`: Fordert den Benutzer auf eine Eingabe zu machen und liest dann ein einzelnes Zeichen.

```
?- get0(N).
|: a
N = 97
Yes
12 ?- get0(N1), get0(N2).
|: ab
N1 = 97
N2 = 98
Yes
```

Zeichencode in Atome umwandeln

`atom_chars(`*-Atom, +CharListe*`)` wandelt eine Liste von Zeichencodes (also Zahlen) in das entsprechende Atom um.

```
?- get0(N), atom_chars(Atom,[N]).
|: a
N = 97
Atom = a
Yes
14 ?- get0(N1), get0(N2), atom_chars(Atom,[N1,N2]).
|: ab
N1 = 97
N2 = 98
Atom = ab
Yes
```

Eingabe aus Dateien lesen

`read(`+*Stream, ?Term*`)`

`get0(`+*Stream, ?Char* `)`

Programmteile aus Dateien laden

Regeln ohne Kopf werden Prolog einmal beim Laden der Datei ausgeführt.

Datei `einProgramm.pl`:

```
:- [mehrPraedikate.pl]
```

Beim Laden von `einProgramm.pl` wird automatisch auch `mehrPraedikate.pl` geladen.

Problem: Wenn `einProgramm.pl` und `mehrPraedikate.pl` Prädikate mit dem gleichen Namen (und der gleichen Arität) definieren, dann gibt es einen Konflikt.

$\Rightarrow$ Module

Module

Module unterstützen Programmieren in Teams und generell größere Projekte, indem sie

- lokale Prädikate verstecken und
- Schnittstellen klar definieren.

Module deklarieren

Datei `reverse.pl`

```
:- module(reverse, [reverse/2]).

reverse(Liste,R) :- reverse_acc(Liste,[],R).

reverse_acc([],Acc,Acc).
reverse_acc([H|T],Acc,R) :- reverse_acc(T,[H|Acc],R).
```

- Ein Modul mit dem Namen `reverse` wird angelegt.
- Das Modul enthält die Definitionen für die Prädikate `reverse/2` und `reverse_acc/3`.
- Nur das Prädikat `reverse/2` ist öffentlich, d.h. kann von Prädikaten außerhalb des Moduls verwendet werden.

Modul-Prädikate importieren

`:- use_module(`*+Datei*`)`

Importiert alle öffentlichen Prädikate aus Datei *Datei*.

`:- use_module(`*+Datei, +Liste_zu_importierender_Prädikate*`)`

Importiert nur die Prädikate, die angegeben sind. Das müssen alles öffentliche Prädikate sein.

Aufgaben

- Ändere das Programm `pptree.pl` so, dass es den Parsebaum in die Datei `pptree.out` schreibt, statt ihn auf dem Bildschirm auszugeben.
- Mache ein Modul aus pptree.

Zusammenfassung

Heute haben wir gesehen:

- wie Ausgaben in eine Datei gemacht werden können,
- wie Terme und einzelne Zeichen eingelesen werden können,
- was Module sind und wie sie benutzt werden.

Wenn ihr die Definition/Funktionsweise von eingebauten Prädikaten vergessen habt; wenn ihr wissen wollt, was es sonst noch für eingebaute Prädikate gibt:

- SWI Prolog Dokumentation:
  `http://www.swi-prolog.org`
- Sicstus Prolog Dokumentation:
  `http://www.sics.se/sicstus/`