

# How To

Benjamin Arnold      Felix Hoeborn

26. Februar 2013

# Inhaltsverzeichnis

<b>1</b>	<b>Top-Level</b>	<b>1</b>
1.1	Includes . . . . .	1
1.2	Defines . . . . .	1
1.3	Klassen . . . . .	1
1.4	Funktionsdefinition . . . . .	1
<b>2</b>	<b>Low-Level</b>	<b>1</b>
2.1	Expressions . . . . .	1
2.1.1	Operationen . . . . .	1
2.1.2	Funktionsaufrufe . . . . .	2
2.1.3	Set . . . . .	2
2.1.4	Terme . . . . .	2
2.2	Terme . . . . .	2

# 1 Top-Level

Diese Sprache hat auf oberster Ebene 4 Bestandteile.

## 1.1 Includes

`include 'Name der zu includierenden Datei' ;`

## 1.2 Defines

Es können globale Variablen definiert werden.

`< type > id ;` für nicht initialisierte Variablen.

`< type > id = term ;` für nicht initialisierte Variablen.

## 1.3 Klassen

`id {`

Innerhalb einer Klasse können Definitionen (siehe 1.2) stehen.

Konstruktoren können nach folgendem Muster definiert werden:

```
cons id ( < type > id ( , < type > id , ... ) )
{
  expr2.1 und/oder define1.2
}
```

`}`

## 1.4 Funktionsdefinition

Funktionen werden wie folgt definiert:

```
< type > id ( < type > id ( , < type > id , ... ) )
{
  expr2.1 und/oder define1.2
}
```

# 2 Low-Level

## 2.1 Expressions

### 2.1.1 Operationen

Operationen haben immer den gleichen Aufbau.

`term2.2 Operant expr2.1`

Operanten sind `+` , `-` , `*` , `/` , `%` , `is` , `smaller` und `bigger`

### 2.1.2 Funktionsaufrufe

Es gibt 3 Arten von Funktionsaufrufen:

call **call** *id*( *id* ( , *id* , ... ) )

new **new** *id*( *id* ( , *id* , ... ) )

destroy **destroy** *id*

### 2.1.3 Set

*id* = *term* ;

### 2.1.4 Terme

Für eine *expr* kann auch ein *term*<sub>2.2</sub> eingesetzt werden.

## 2.2 Terme

*expr* ( *expr* )

*id* Zeichenketten mit großen und kleinen Buchstaben sowie Sterne

*number* ganze Zahlen

*null* **null**

*boolean* **true** oder **false**