

# Tutorium - Informatik 1

---

Montags 16 st.

Jules Kreuer, VL bei Prof. Grust

## 🚩 Reale Namen im Forum und auf Discord [ ⚠️ YOUR ACTION REQUIRED bis 4. Dezember 2020 ]

■ Informatik 1 (WS20/21)



**Torsten Grust** Professor

2 🗨️ 5h

Servus zusammen,

dieses Wintersemester ist (leider) anders und persönliche *face-to-face* Kommunikation fällt wohl größtenteils flach. Anstelle des persönlichen Kontaktes tritt elektronische Kommunikation, etwa Videokonferenzen, unser Discord-Chat oder auch dieses Forum.

Aufgrund der zentralen Kommunikationsrolle von Discord und Forum für unseren Vorlesungs- und Übungsbetrieb, **werden wir dieses Forum und auch Discord jetzt auf die Nutzung von realen Namen umstellen**. Jetzt und in den kommenden Wochen/Monaten mit „Foo Bar“, „Teggy“, „Tom Nook“, „Awesome Dude“, „FN2187“, o.ä. zu verbringen, wird dieser Rolle kaum gerecht und lässt einen im Unklaren, mit wem man eigentlich umgeht. Das implementiert einerseits eine Policy des WSI. Andererseits erinnert die Nutzung realer Namen daran, dass wir hier tatsächlich mit Kommilitonen umgehen (und nicht etwa virtuellen NPCs). Die meisten von euch nutzen bereits eure realen Vor- und Zunamen; herzlichen Dank dafür.

Wir bitten wir daher jede(n) unter euch **bis spätestens Freitag, 4. Dezember**,

1. **[required]** auf dem **Discord-Server** der *Informatik 1* mittels des Kommandos `/nick <name>` den *Nickname* auf euren *realen Namen (Vor- und Zunamen)* zu setzen (diese Nicknames sind server-lokal und beeinflussen nicht euer Auftreten auf anderen Discord-Servern),
2. **[required]** in euren **Forums-Einstellungen** im Feld *Name* euren *realen Namen (Vor- und Zunamen)* einzutragen (der Benutzername `@<name>` bleibt dabei unverändert) und
3. **[optional]** ein Portrait von euch mittels *Change Avatar* (Discord) bzw. unter *Profilbild* (Forum) einzutragen. Ihr findet unten einen Screenshot, der zeigt, wie das derzeit in meinen Forums-Einstellungen aussieht.

Achtung 🙄: Nach dem 4. Dezember werden wir aus den uns vorliegenden Anmeldedaten reale Namen konstruieren und in den Einstellungen von Discord und Forum eintragen.

Schon jetzt besten Dank, beste Grüße,  
—Torsten Grust



30. Nov.

1 / 2

30. Nov.

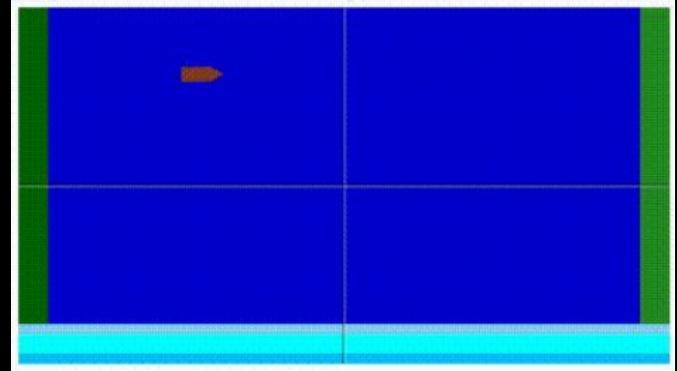
vor 5h



<https://forum-db.informatik.uni-tuebingen.de/t/reale-namen-im-forum-und-auf-discord-your-action-required-bis-4-dezember-2020/8857>

# ÜB 2

Einfache Funktionen &  
Animationen



# Schwierigkeiten bei ÜB 2

- Signaturen von den ÜB übernehmen!
- Wenn gilt  $\forall a, b f(a) = f(b)$  mit  $a \neq b$  ist  $f$  eine konstante Funktion
  - $\Rightarrow$  Konstantendefinition
- Keep it simple, stupid!
  - viele sehr 'komplizierte' Abgaben
  - Interpolate wird nicht gebraucht (drift does the job)

# ÜB 3

## Reduktion und Vereinfachung

### Informatik 1

Forum: <https://forum-db.informatik.uni-tuebingen.de/c/ws2021-info1>

### Übungsblatt 3 (25.11.2020)

Abgabe bis: Mittwoch, 2.12.2020, 14:00 Uhr



Relevante Videos: bis einschließlich Informatik 1 - Chapter 03 - Video #015.

<https://tinyurl.com/Informatik1-WS2021>

Sprachebene „Die Macht der Abstraktion — Anfänger“

#### Aufgabe 1: [12 Punkte]

Reduziert die unten angegebenen Scheme-Ausdrücke (a) bis (c) nach den Reduktionsregeln [·] für Scheme, die in der Vorlesung besprochen wurden.

**Wichtig:** Notiert *alle* Reduktionsschritte genau wie in den Beispielen der Vorlesung zu *Chapter 3* auf den *Folien 5 bis 7* (nutzt dazu Racket-Kommentare).

- (a) Reduziert zunächst den **arithmetischen Ausdruck**

```
(* 2 (+ 20 1))
```

Reduziert anschließend den semantisch äquivalenten Ausdruck

```
(+ (+ 20 1) (+ 20 1))
```

Vergleicht die Anzahl der jeweils benötigten Reduktionsschritte.

- (b) Gegeben die folgenden **Konstanten- und Funktionsdefinitionen:**

```
(define pi 3.1415)  
(define perimeter  
  (lambda (r)
```

# ÜB 3

Reduktion und Vereinfachung

Signaturen  
Conditionals  
[[Reduktion]]

# Signaturen

- Für Funktionen (ggf für Konstanten)
- So “streng” wie möglich, so locker wie nötig
- `(: funktion (eingabe1 eingabeN -> ausgabe) )`
- Custom Signaturen mittels one-of
  - erzeugt Signatur
  - braucht explizite Werte (keine weiteren Signaturen)
  - Geht `(: funktion (one-of 1 2 3 #t #f) -> boolean)`
  - Geht nicht: `(: funktion (one-of natural boolean) -> boolean)`

# Conditionals

- Fallunterscheidungen

- if: `(if a b c)` ← sonst c  
↑  
boolean  
↑  
wenn a == #t

- cond: `(cond a b wenn a == #t`  
`a2 b2 wenn a2 == #t`  
`...`  
`an bn`  
`else bn+1)` wenn kein `a* == #t`, optional,  
Fehler wenn kein `a* == #t` und kein `else`

# Prädikate

Untersuchung einer Eigenschaft -> Boolean

=

< / > / <= / >=

string=?

boolean=?

zero?

odd? / even?

positive? / negative?

Später: (define-record-procedures ...) mit Prädikat

# And / Or

- (and  $b_1 \dots b_n$ )  $\rightarrow$  #t wenn alle  $b^* == \#t$   
#f sonst
- (or  $b_1 \dots b_n$ )  $\rightarrow$  #t wenn ein  $b^* == \#t$   
#f sonst



# [[Reduktion]]

- Herunterbrechen von Ausdrücken / Funktionen / usw
- Universell anwendbar
- Eindeutig

Die **Auswertung** in Scheme  $e \rightsquigarrow \llbracket e \rrbracket$  **reduziert** einen Ausdruck  $e$  zu seinem Wert  $\llbracket e \rrbracket$  (auch:  $\llbracket e \rrbracket$  ist die **Semantik** von  $e$ ).

Definiere ( $\stackrel{\text{def}}{=}$ )  $\llbracket e \rrbracket$  je nach Art des Ausdrucks  $e$ :

**1 Literal**  $l$  ( $1$ ,  $\#t$ , `"abc"`,  $\#<\text{procedure:}\otimes>$ , ...):  $[\text{eval\_lit}]$

$\llbracket l \rrbracket \stackrel{\text{def}}{=} l$

---

**2 Lambda-Abstraktion** (`lambda` ( $v_1 \dots v_n$ )  $e$ ):  $[\text{eval\_}\lambda]$

$\llbracket (\text{lambda } (v_1 \dots v_n) e) \rrbracket \stackrel{\text{def}}{=} (\text{lambda } (v_1 \dots v_n) e)$

---

**3 Identifier**  $id$ :  $[\text{eval\_id}]$

$\llbracket id \rrbracket \stackrel{\text{def}}{=} \llbracket e \rrbracket$  , falls (`define`  $id$   $e$ ) gilt

Definiere  $\llbracket e \rrbracket$  je nach Art des Ausdrucks  $e$ :

### ▣ Applikation $(f e_1 \dots e_n)$

$\llbracket (f e_1 \dots e_n) \rrbracket$

$$\stackrel{\text{def}}{=} \begin{cases} \llbracket \llbracket e_1 \rrbracket \otimes \dots \otimes \llbracket e_n \rrbracket \rrbracket, & \text{falls } f = \# \langle \text{procedure:} \otimes \rangle & \text{[apply\_prim]} \\ \llbracket e \{v_1 \rightarrow \llbracket e_1 \rrbracket, \dots, v_n \rightarrow \llbracket e_n \rrbracket\} \rrbracket, & \text{falls } f = (\text{lambda } (v_1 \dots v_n) e) & \text{[apply\_}\lambda\text{]} \\ \llbracket (\llbracket f \rrbracket e_1 \dots e_n) \rrbracket, & \text{sonst} & \text{[apply]} \end{cases}$$

- $e \{v_1 \rightarrow e_1, \dots, v_n \rightarrow e_n\}$ : In  $e$  wird Identifier  $v_i$  durch Ausdruck  $e_i$  ersetzt.

Beispiel:  $(+ (* 4 y) x) \{x \rightarrow 2, y \rightarrow 10\} = (+ (* 4 10) 2)$ .

Aus: Informatik 1 - bei Prof. Torsten Grust

# ÜB 3

Reduktion und Vereinfachung

Live  
Programming