

# Fallunterscheidungen

## Einführung in die Programmierung

Johannes Brauer

17. Januar 2020

## Fallunterscheidungen aus der Mathematik

Zwei Beispiele

### Definition der Fakultät

$$n! = \begin{cases} 1, & \text{falls } n = 0 \\ n \cdot (n-1)!, & \text{falls } n \geq 1 \end{cases}$$

### Definition der Fibonacci-Folge

Die Fibonacci-Folge  $f_1, f_2, f_3, \dots$  ist wie folgt definiert:

$$f_1 = 1 \tag{1}$$

$$f_2 = 1 \tag{2}$$

$$f_n = f_{n-1} + f_{n-2} \text{ für } n > 2 \tag{3}$$

## Bedingte Funktionen

### Vergleichsausdrücke und Boolesche Ausdrücke

- Numerische Vergleichsausdrücke der Art

$$a < b, x \geq y \text{ oder } r < s < t$$

werden in Racket so aufgeschrieben:

$$(< \text{ a } \text{ b } ) \text{ } (>= \text{ x } \text{ y } ) \text{ } (< \text{ r } \text{ s } \text{ t } )$$

- Die Auswertung ergibt `#true` oder `#false`.
- Den Racket-Ausdruck `(< r s t)` kann man als Abkürzung für

$$(\text{and } (< \text{ r } \text{ s } ) \text{ } (< \text{ s } \text{ t } ))$$

betrachten.

- Neben `(and ...)` stehen `(or ...)` und `(not ...)` zur Verfügung. Die Anzahl der Argumente von `and` und `or` ist dabei beliebig groß.

## Anwendungsbeispiel

- Wir nennen eine Funktion *bedingt*, wenn für die Ermittlung ihres Resultats eine *Fallunterscheidung* erforderlich ist.
- Beispiel: Ein Hersteller für Speicherchips verkauft die Chips nach folgender Preisstaffel:

Stückzahl	Stückpreis [€]
$\leq 1000$	15,00
$> 1000$ und $\leq 10000$	12,50
$> 10000$	9,75

Gesucht: Funktion, die aus der Stückzahl den Stückpreis ermittelt

## Racket-Pseudofunktion für Fallunterscheidungen

```
(cond
  [frage antwort]
  ...
  [frage antwort])

(cond
  [frage antwort]
  ...
  [else antwort])
```

- Jede *frage* muss ein boolescher Ausdruck sein.
- Jede *antwort* ist ein beliebiger Racket-Ausdruck.
- Das Resultat der cond-Funktion ist die *antwort* der ersten *frage*, deren Auswertung `#true` liefert.
- In der linken Variante von `cond` muss die Auswertung mindestens einer *frage* `#true` liefern.

## Entwurf bedingter Funktionen – Regel 6

- Für den Entwurf einer bedingten Funktion sind in der Problembeschreibung alle zu unterscheidenden Fälle zu identifizieren.
- Für die gemäß Regel 3 erforderlichen Beispiele ist für jeden identifizierten Fall mindestens ein Beispiel aufzuschreiben.
- Zusätzlich sind die Grenzfälle (Intervallgrenzen) zu beachten.

### Regel 6:

- Für den Funktionsrumpf ist
  - ein `cond`-Skelett mit je einer Frage-Antwort-Kombination für jeden Fall zu formulieren,
  - für jeden Fall die Frage (Bedingung) zu formulieren,
  - für jeden Fall der Racket-Ausdruck für die Berechnung der Antwort zu ermitteln.

## Entwurf der Funktion `stueckpreis`

### Das `cond`-Skelett

Für das Beispiel des Chipherstellers ergibt sich folgendes Funktionsskelett:

```
;; berechnet aus einer gegebenen Stueckzahl  
;; den Stueckpreis gemaess Preisstaffel  
(define stueckpreis  
  (lambda [stueckzahl]  
    (cond  
      [... ...]  
      [... ...]  
      [... ...])))
```

### Formulierung der Fragen

Aus der Tabelle für die Preisstaffelung ergeben sich folgende Bedingungen/Fragen:

```
;; berechnet aus einer gegebenen Stueckzahl  
;; den Stueckpreis gemaess Preisstaffel  
(define stueckpreis  
  (lambda [stueckzahl]  
    (cond  
      [(and (>= stueckzahl 0)  
            (<= stueckzahl 1000)) ...]  
      [(and (> stueckzahl 1000)  
            (<= stueckzahl 10000)) ...]  
      [(> stueckzahl 10000) ...])))
```

Für die Beispiele sollten die Grenzfälle 0, 1000 und 10000 sowie Werte aus dem Innern der Intervalle (z. B. 500, 2000, 20000) verwendet werden.

### Formulierung der Antworten

Die Ausdrücke für die Berechnung der Antworten ergeben sich direkt aus der Tabelle für die Preisstaffelung:

```
;; berechnet aus einer gegebenen Stueckzahl  
;; den Stueckpreis gemaess Preisstaffel  
(define stueckpreis  
  (lambda [stueckzahl]  
    (cond  
      [(and (>= stueckzahl 0)  
            (<= stueckzahl 1000)) 1500]  
      [(and (> stueckzahl 1000)  
            (<= stueckzahl 10000)) 1250]  
      [(> stueckzahl 10000) 975])))
```

### Vereinfachung der Bedingungen

Nachdem die Funktion getestet wurde, können die Bedingungen vereinfacht werden:

```
(define stueckpreis
  (lambda [stueckzahl]
    (cond
      [(<= stueckzahl 1000) 1500]
      [(<= stueckzahl 10000) 1250]
      [else 975])))
```

(Die vollständige Funktion **stueckpreis** steht in Moodle zur Verfügung.)  
(Aufgaben zu bedingten Funktionen)