Container füllen

Schrittweise Erarbeitung
der einfachen
Fülle - Funktion

Container füllen

einfachste Variante:

Fülle jeweils das größte noch passende Stück ein.

Container füllen

Was müssen wir kennen?

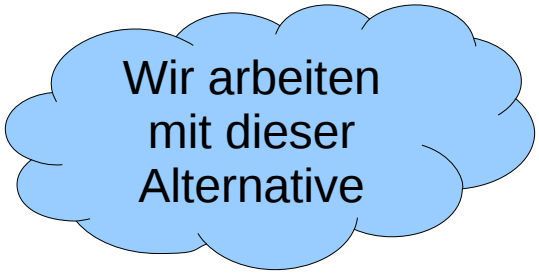
- die Stücke
  Wir definieren eine Liste:

  '(30 30 30 30 20 20 20 20)

  Datenstruktur
  Liste
- den Container
  Auch dazu verwenden wir ein Liste,
  die zunächst leer ist.
- die Zielgröße, also das Füllvermögen.
  Für sie benötigen wir allein eine Zahl: 80.

Container füllen

Was müssen wir kennen?



- die Stücke
  Wir definieren eine Liste:

  '(30 30 30 30 20 20 20 20)

  Datenstruktur
  Liste

- den Container
  Auch dazu verwenden wir ein Liste,
  die am Kopf die die Zielgröße enthält,
  sonst aber zunächst leer ist.

Container füllen

1. Schritt: Definition des Funktionskopfes

```
(define
  (fuelle stuecke container)
  'undefiniert)
```

Testaufruf:

```
(fuelle
  '(30 30 30 30 20 20 20 20)
  '(80 30 30 20)
  )
```

Container füllen

Was kann passieren?

[Das ist die Frage nach den Abfragen im cond]

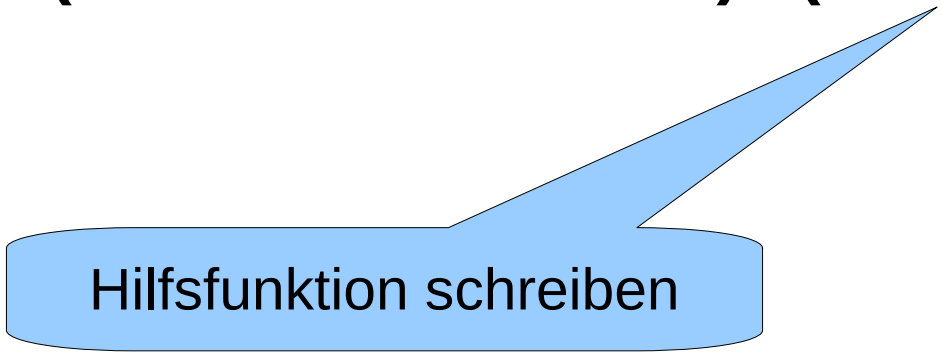
- die Stücke passen genau
  Dazu muss die Summe aller Stücke im Container gleich der Zielgröße (am Kopf der Containerliste) sein.
- Um diese Prüfung zu erledigen, schreiben wir eine Hilfsfunktion, der wir die Containerliste und die Zielgröße übergeben.

Container füllen

Hilfsfunktion exakt-voll?

Sie prüft, ob der Container die Zielgröße erreicht hat.

```
(define
  (exakt-voll? container)
  (= (first container) (fuellung container))
  )
```



Container füllen

Hilfsfunktion fuellung

```
define
  (fuellung container)
  ;;; einfache Loesung mit apply
  (apply + (rest container)))
```

apply wendet eine Funktion
auf alle Elemente einer Liste an

```
;;; Alternativ Hilfsfunktion summe
einsetzen
```

Container füllen

2. Schritt: Erfolgsfall testen

```
(define
  (fuelle stuecke container)
  (cond
    ((exakt-voll? container)
     container)
    (else
     'undefiniert)))
```

Container füllen

Der Test:

```
(fuelle
  '(30 30 30 30 20 20 20 20)
  '(80 30 30 20)
  )
```

liefert nun wie erwartet die gewünschte Lösung.

Container füllen

Was kann passieren?

- die Stücke passen genau
- der Container wird zu voll
  Das ist der Fall, wenn das neue Stück mit der Summe aller Stücke im Container größer als die Zielgröße ist.
- Es ist einfach, die zugehörige Hilfsfunktion zu schreiben.

Container füllen

Hilfsfunktion zu-voll?

Sie prüft, ob der Container mit dem Stück die Zielgröße überschreitet.

```
(define
  (zu-voll? stueck container)
  (<
    (first container)
    (+ stueck (fuellung container)))
  )
```

Container füllen

3. Schritt: Den Fall *"zu voll"* bearbeiten:
Weiter versuchen ohne das Stück.

```
(define
  (fuelle stuecke container zielgroesse)
  (cond
    ((exakt-voll? container zielgroesse)
     container)
    ((zu-voll? Container zielgroesse)
     (fuelle (rest stuecke) container))
    (else
     'undefiniert)))
```

Container füllen

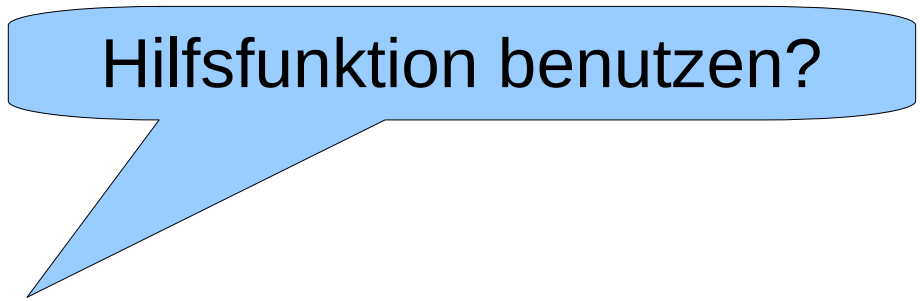
Was kann passieren?

- die Stücke passen genau
- der Container wird zu voll
- das aktuelle Stück passt noch in den Container
  Dieser Fall muss aber nicht durch eine weitere Abfrage geprüft werden, da er den "Normalfall" darstellt. → else

Container füllen

4. Schritt: Das Stück geht noch hinein.

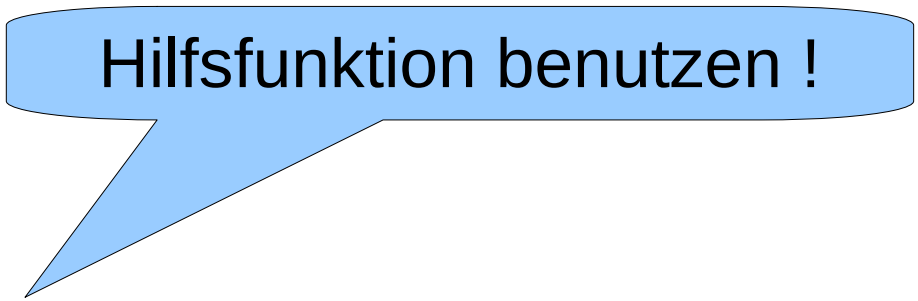
```
(define
  (fuelle stuecke container)
  (cond
    ...
    ...
    (else
     (fuelle
      (rest stuecke)
       (cons
        (first container)
        (cons (first stuecke container)))
  ))
```



Container füllen

4. Schritt: Das Stück geht noch hinein.

```
(define
  (fuelle stuecke container)
  (cond
    ...
    ...
    (else
     (fuelle
      (rest stuecke)
      (fuelle-ein
       (first stuecke)
       container)))
  ))
```



Container füllen

Hilfsfunktion fuelle-ein

Füllt ein Stück in den Container ein.

```
(define
  (fuelle-ein stueck container)
  (cons
   (first container)
   (cons stueck (rest container)))
  )
```

Container füllen

Was kann passieren?

- die Stücke passen genau
- der Container wurde zu voll
- das aktuelle Stück passt noch in den Container
  Dieser Fall muss aber nicht durch eine weitere Abfrage geprüft [illegible] "Normalfall" darstellt. →

Typischer Fehler:
Beim Abbau von Listen muss auch der Fall einer leeren Liste berücksichtigt werden.

Container füllen

Was kann passieren?

Warum diese Reihenfolge?

- die Stücke passen genau
- es gibt keine Stücke mehr
- der Container wurde zu voll
- das aktuelle Stück passt noch in den Container
- der Container ist noch nicht voll

Container füllen

```
(define
  (fuelle stuecke container zielgroesse)
  (cond
    ((exakt-voll? ...)
    ((null? stuecke) #f)
    ((exakt-voll? ...) ..)
    ((zu-voll? ...)
    )else    ...)))
```