

6 Fallstudie: Kevin allein zu Hause

6.1 Spielbeschreibung

Kevin's Eltern machen sich ein paar nette Stunden ohne Kevin.

Daher will Kevin möglichst die gesamte Zeit damit verbringen, Papas Videos anzuschauen.

Aufgabe:

Hilf Kevin, die Videos so auszusuchen, dass möglichst wenig Zeit übrig bleibt.

6.2 Wiederholte Berechnungen: Der let-Ausdruck

```
(define square-sum  
  (lambda (x y)  
    (* (+ x y) (+ x y))))
```

wiederholt die Auswertung von (+ x y).

Verbesserung durch benanntes Zwischenergebnis:

```
(let ((sum (+ x y))) (* sum sum))
```

Format: (let ((*variable* *expression*₁)) *expression*₂)

- erster Ausdruck *expression*₁ wird ausgewertet
- Wert des ersten Ausdrucks wird für Variable im zweiten Ausdruck eingesetzt
- zweiter Ausdruck *expression*₂ wird ausgewertet und liefert Wert des gesamten Ausdrucks

⇒ *variable* ist *lokale Variable*, die nur in *expression*₂ bekannt ist!

6.3 Wiederholte Berechnungen vermeiden

```
(define square-sum
  (lambda (x y)
    (let ((sum (+ x y)))
      (* sum sum))))

(square-sum 4 3)
=> ((lambda (x y) (let ((sum (+ x y))) (* sum sum))) 4 3)
=> (let ((sum (+ 4 3))) (* sum sum))
=> (let ((sum 7)) (* sum sum))
=> (* 7 7)
=> 49
```

6.4 Verwendung von Let-Ausdrücken

- Vermeiden von wiederholten Berechnungen
- Definition von benannten Zwischenergebnissen

Let-Ausdrücke sind syntaktischer Zucker

- erleichtern das Programmieren (Lesbarkeit)
- können durch Kombination anderer Ausdrücke beschrieben werden

Alternative Definition von let

$(\text{let } ((x \langle \text{expression} \rangle_1)) \langle \text{expression} \rangle_2)$

\equiv

$((\text{lambda } (x) \langle \text{expression} \rangle_2) \langle \text{expression} \rangle_1)$

6.5 Let*-Ausdrücke

Ein geschachtelter let-Ausdruck

```
(let ((x (- a b)))  
  (let ((y (- b c)))  
    (let ((z (- c d)))  
      (+ (* x y) (* y z) (* z x))))))
```

kann zur besseren Lesbarkeit als let*-Ausdruck geschrieben werden:

```
(let* ((x (- a b))  
      (y (- b c))  
      (z (- c d)))  
  (+ (* x y) (* y z) (* z x)))
```

MANTRA

Mantra #9 (Wunschdenken, Top-Down-Design)

Verschiebe Probleme, die nicht sofort lösbar sind, in noch zu schreibende Prozeduren. Lege für diese Prozeduren Beschreibung und Vertrag fest, aber schreibe sie später.

Mantra #10 (lokale Variable)

Benenne Zwischenergebnisse mit lokalen Variablen.