Inhaltsverzeichnis

1 Scheme

Übersicht über funktionale Programmiersprachen:

- LISP
  - Common Lisp
  - Scheme (Racket)
  - Clojure
- ML Familie
  - SML (*Standard ML*)
  - OCaml
  - F#
- Haskell

LISP (*LISt Processing*) wurde 1958 von John McCarthy entwickelt und ist inspiriert durch das Lambda-Kalkül.

Lisp ist *homoikonisch*, (sebst-abblildbar oder selbst-repräsentierbar), d.h. Programme dieser Sprache sind gleichzeitig Datenstrukturen derselben Sprache. Dadurch ist es einfach, Programme zu schreiben, die Programme schreiben. In Lisp ist jedes Programm eine Liste.

Im Zentrum stehen Funktionen und die Anwendung von Funktionen, ohne Seiteneffekte (*referenzielle Transparenz*) und mit Funktionen als gleichberechtigten Datenobjekten (*Funktionen höherer Ordnung*) und Funktionsimplementierungen für verschiedene Typen (*Polymorphismus*). Dies führt insgesamt zu kurzen und klaren Programmen, die einfacher wartbar sind und schneller erstellt werden können.

Aufgaben werden anhand von *Dekomposition* und*Komposition* gelöst: ein Problem wird in kleinere Teilprobleme zerlegt und anschliessend definiert, wie die kleinsten (atomaren) Probleme zusammengesetzt werden müssen.

1.1 Beispiel: Java vs Scheme

Berechnung des Mittelwerts von Quadratzahlen von 0 bis n-1 in Java:

```java
// Mittelwert der Quadratzahlen 0..n-1
public class MeanOfSquares {
  public static void main(final String[] args) {
    final int count = Integer.parseInt(args[0]);
    double sum = 0;
    for (int i = 0; i < count; i++) {
      sum = sum + i * i;
    }
    System.out.println(sum / count);
  }
}
```

Das gleiche in Scheme

```scheme
; Mittelwert der Quadratzahlen 0..n-1
(define (mean-of-squares n)
  (/ (apply + (build-list n sqr)) n))
```

1.2 Formen und Funktionen

1.2.1 Primitive Ausdrücke: Werte

Zahlen sind selbstauswertend: die Werte der Ziffern sind die Zahlen, die sie bezeichnen. Scheme unterscheidet zwischen:

- Ganzen Zahlen (integer): `6`, `42`
- Rationalen Zahlen (rational): `-¾`, `6.9`
- Reelle Zahlen (real): `+inf.0`
- Irrationale Zahlen, bsp. π: `#i3.141592653589793`
- Komplexe Zahlen (complex), bsp: $\sqrt{-2}$: `#i0+1.4142135623i`

Boolesche Werte (mit alternativen Darstellungen):

- `true`, `#true`, `#t`
- `false`, `#false`, `#f`

In Scheme eingebaute Operatoren werden PRIM-OPS (*primitive operations*) genannt. Einige Operatoren für ganze Zahlen:

- `+`

- `-`
- `*`
- `/`
- `abs`
- `min`
- `gcd`
- …

Einige Operatoren für reelle und komplexe Zahlen:

- `exp`
- `sin`
- `log`
- `sqr`
- `sqrt`
- …

1.2.2 Form

Alles, was in Scheme eingegeben wird, ist eine *Form*: