Vertiefung Funktionale Programmierung: Clojure

Programmierübung 1

Philipp Körner

Institut für Informatik
Heinrich-Heine-Universität Düsseldorf

15. April 2024

Kombinationen

Implementieren Sie eine Funktion, die zu einer Sequenz alle Kombinationen von zwei ihrer Elemente bestimmt. Duplikate in der Eingabe müssen nicht betrachtet werden.

```
user=> (combinations [1 2 3])
#{(1 2) (1 3) (2 3)}
user=> (combinations [1 2 3 4])
#{(1 2) (1 3) (1 4) (2 3) (2 4) (3 4)}
```

Lauflängenkodierung

Die Lauflängenkodierung ist ein verlustfreies Kompressionsverfahren, das besonders gut bei vielen Wiederholungen desselben Symbols funktioniert. Dazu wird das auftretende Symbol mit der entsprechenden Häufigkeit übermittelt anstatt das Symbol entsprechend häufig zu senden.
Beispielsweise kodiert man `[1 3 3 3 3 3 3 3 7]` als `[1 1 7 3 1 7]` („ein mal 1, sieben mal 3, ein mal 7“).
Implementieren Sie eine Funktion `compress`, die eine beliebige Sequenz nach den Lauflängen kodiert.
Implementieren Sie eine Funktion `decompress`, die die Lauflängenkodierung wieder expandiert.

Oscilrate, 4clojure Nr. 98

Schreiben Sie ein oszillierendes iterate: Die Funktion nimmt einen Startwert und beliebig viele Funktionen. Es soll eine (lazy) Sequenz von Funktionswerten in der Reihenfolge der Funktionen zurückgegeben werden, beginnend mit dem Startwert. Ist die Funktionssequenz abgearbeitet, beginnt diese erneut.

```
user=> (take 3 (oscil 3.14 int double))
[3.14 3 3.0]
user=> (take 5 (oscil 3 #(- % 3) #(+ 5 %)))
[3 0 5 2 7]
user=> (take 12 (oscil 0 inc dec inc dec inc))
[0 1 0 1 0 1 2 1 2 1 2 3]
```