Skizze Aufgabe 1

Team: <03>, <Jan Dennis Bartels, Patrick Steinhauer>

Aufgabenaufteilung:

1. Bisher beide für alles. Skizze zusammen erarbeitet und angefangen mit der ADT Liste herum zu probieren.
2. Aufgaben wurden zusammen bearbeitet.

Quellenangaben: < ---------- >

Begründung für Codeübernahme: <Keine Code Übernahme.>

Bearbeitungszeitraum: <02.10.2014, 9 Stunden ca. für die Bearbeitung der Skizze + Coding der Aufgabe>

Aktueller Stand: Source Dateien alle fertig bearbeitet und fertig gestellt.

Änderungen der Skizze: Die Signaturen wurden eingefügt.

Skizze:

Teil 1 Fragestellungen, Hilfen, Überlegungen :

Als erstes bearbeiten wir den Teil mit der ADT Liste.

Hierbei gibt es erst einmal ein paar generelle Sachen die zu klären sind, bezüglich Erlang selbst und wie verschiedene Sachen dort aussehen, da wir bisher ja erst in Erlang eingestiegen sind.

ADT Liste :

- Wie sieht eine Liste aus?
- Wie könnte man diesen Entwurf einer Liste realisieren?
- Was muss möglicherweise beachtet werden?

Eine Liste besteht in unserem Entwurf aus den „{ }“.

Elemente einer Liste werden hier durch ein Komma getrennt. Elemente einer Liste können beliebige Elemente sein, wie z.B. Atome, Zahlen, weitere Listen etc.

Eine Liste, wie wir bisher wissen besteht aus einem Head und einem Tail.

Der Head ist jeweils immer das Element, welches an erster Stelle steht. Im Tail befinden sich alle weiteren Elemente der Liste.

Hierbei ist ebenfalls ein Index für jedes Element vorhanden, welches die Liste beinhaltet. Weiterhin wird eine Liste, die so aussieht „{ }“ leere Liste genannt. Die leere Liste dient oft bei einer Rekursion für den Rekursionsabbruch.

create: $\emptyset \rightarrow list$

isEmpty: $list \rightarrow bool$

laenge: $list \rightarrow int$

insert: $list \times pos \times elem \rightarrow list$

delete: $list \times pos \rightarrow list$

find: $list \times elem \rightarrow pos$

retrieve: $list \times pos \rightarrow elem$

concat: $list \times list \rightarrow list$

Create :

- Das create soll eine Liste erstellen.
- Hierbei ist zu klären, wie man das Erstellen definiert.
    - Das create soll beim Erstellen der Liste einfach eine leere Liste zurückgegeben wird, mit der man weiter arbeiten kann.
    - Die zweite Überlegung ist, dass man die Struktur der Liste versucht genauer abzubilden.
- Das realisieren der ersten Überlegung würde dann so aussehen:
    - Funktionsaufruf → leere Liste

- Der Index bei der Liste beginnt bei 1.

isEmpty :

- Die Funktion isEmpty soll prüfen, ob die Liste unserem create entspricht.
- Zurückgegeben wird true oder false.

Insert :

- Fügt ein Element zu der Liste hinzu.

- Destruktivität (nicht destruktiv) beachten! Wenn ein Element an einer vorhandenen Stelle eingefügt wird, verändern sich die Positionen der folgenden Elemente.
- Wenn es die Position größer der Listenlänge ist, einfach die vorhandene Liste zurückgeben.

Delete :

- Soll ein vorhandenes Element aus der Liste löschen.
- Hierbei ebenfalls die Position beachten, wenn es eine Position ist welche nicht vorhanden ist, Fehler ignorieren und alte Liste zurückgeben.

Find :

- Find sucht das erste Vorkommen eines Elementes und gibt dessen Position zurück.
- Wenn das Element nicht gefunden wird 0 zurückgegeben.

Retrieve :

- Retrieve sucht ein Element an einer angegebenen Position.
- Wenn es diese Position nicht geben sollte, soll -1 zurückgegeben werden.

Concat :

- Fügt zwei Listen zusammen, wobei wieder eine neue Liste entsteht.

ADT stack :

createS: ∅ → stack

push: stack × elem → stack

pop: stack → stack

top: stack → elem

isEmptyS: stack → bool

- Beim Stack ist zu beachten, dass er auf der Liste aufbaut und dass hier gilt „Last In First Out“
- Des Weiteren ist zu beachten, das beim push vorne ein Element angehängt wird und, dass das dieses Element zum neuen Top wird.
- Weiterhin gilt, dass beim pop vorne beim Index 1das Element entfernt wird und hier wird das folgende Element zum neuen Top.

ADT queue :

createQ: ∅ → queue

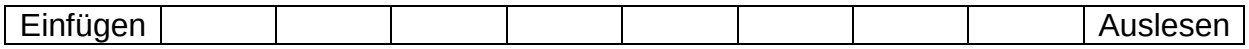
front : queue → elem (Selektor)

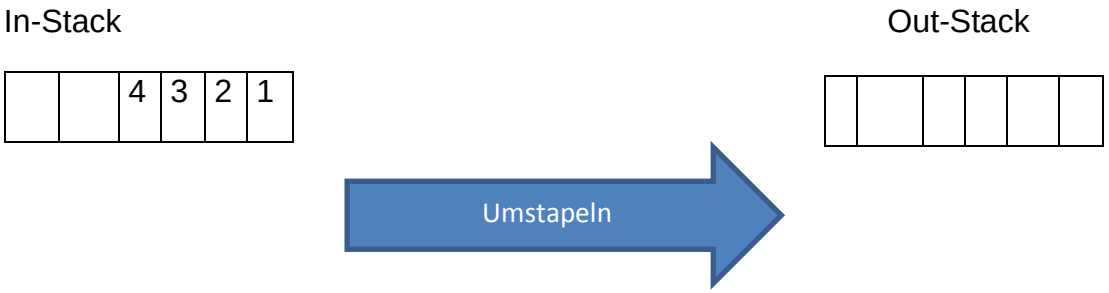
enqueue : queue × elem → queue

dequeue : queue → queue (Mutator)

isEmptyQ : queue → bool

- Die ADT queue baut auf dem ADT stack auf.
- Umgesetzt wird die queue mit einem IN und einem OUT stack
    - Eingefügt wird hier nur am Anfang und Ausgelesen wird am Ende.
- Es funktioniert nach dem Prinzip „First in First out“.
- Die queue kann nur umgestapelt werden wenn der Out stack leer ist





ADT Array :

initA: ∅ → array

setA: array × pos × elem → array

getA: array × pos → elem

lengthA: array → pos

- Der ADT Array wird ebenfalls auf der vorhandenen ADT Liste aufgebaut.
- Arrays beginnen im Gegensatz zur Liste beim Index 0.
- Arrays können beliebig groß sein.
- Arrays arbeiten destruktiv, was bedeutet, dass wenn ein Element an einer bestimmten Position eingefügt wird, dann wird das dort vorhandene Element überschrieben.
- Wenn man auf ein nicht beschriebenen Index des Arrays zugreift, wird eine null zurückgegeben.