4. Arbeitsblatt

Ausgabe: 10.12.98
Bearbeitung: Zwei Termine

Shi Hui
Kolyang
Axel Lankenau
Christoph Lüth
Lutz Zegartowski

Lektüre

Literatur zu abstrakten und polymorphen Datentypen.

Leitfragen

1. Wie kann bzw. sollte ein abstrakter Datentyp (ADT) in einer abstrakten (programmiersprachenunabhängigen) Form definiert werden?
2. Welche Konzepte gibt es in *Hugs*, mit denen sich ADTs realisieren lassen?
3. Welche Vorteile haben ADTs?
4. Was ist ein polymorpher Datentyp?

Anwendungsaufgaben (die in der Gruppe entwickelt und analysiert werden sollen)

1. *Queue*
   Betrachte die in Abschnitt 12.3 aufgezeigten Queue-Modelle. Erkläre das Verhalten der jeweiligen Modelle bei Durchführung folgender Operationen in der gegebenen Reihenfolge: `add 2`, `add 1`, `remove item`, `add 3`, `remove item`, `add 1`, `add 4`, `remove item`, `remove item`. Gib eine Signatur für den ADT *einer* Queue und eine Implementierung der Funktionen `addQ` und `remQ` an.
2. *Dequeue*
   Eine Dequeue ist eine Queue, der Elemente an beiden Enden hinzugefügt oder entnommen werden können. Definiert einen ADT für eine Dequeue. Überlegt Euch dazu zunächst die Signaturen von Operationen wie `emptyDQ`, `isemptyDQ`, `addDQ` und `remDQ`.
3. *Mengen*
   Der Datentyp *Menge* (`Set`) mit den Operationen `singleton`, `empty` und `union` sei wie folgt definiert:

```
type Set a = [a]
  in
    union :: Set a-> Set a-> Set a,
    singleton :: a-> Set a,
    empty :: Set a

empty = []
singleton x = [x]
union x y = x ++ y
```

- Erweitert den ADT um die Operationen $\in$ (element) und $\subseteq$ (subset). Wie verändert sich dabei die Signatur?
- Wie läßt sich die Funktion `union` korrekt realisieren, d. h. so, daß keine Elemente mehrfach in der Liste repräsentiert werden?
- Benutzt die neuen Operationen, um Gleichheit auf Mengen zu definieren und damit den ADT `Set a` als eine Instanz der Typklasse `Eq` zu deklarieren .