Praktische Informatik 3 WS 10/11

Christoph Lüth
Bernd Gersdorf
Christian Maeder
René Wagner
Dennis Walter
Diedrich Wolter
Simon Ulbricht

1. Übungsblatt

Ausgabe: 04.11.10
Abgabe: 15.11.10

1 *Die Wurzel des Ganzen* *10 Punkte*

Die *ganzzahlige Quadratwurzel* einer Zahl $n \in \mathbb{N}$ ist definiert als die größte Zahl x, deren Quadrat noch kleiner gleich n ist:

$$x^2 \leq n \wedge n < (x+1)^2 \tag{1}$$

In diesem Übungsblatt implementieren wir ein Näherungsverfahren zur Berechnung von x. Die Berechnung soll nur unter Benutzung von ganzzahliger Arithmetik erfolgen und keine Fließkommaarithmetik benutzen.

Das Näherungsverfahren beruht auf dem sogenannten babylonischen Verfahren, und berechnet x durch eine Folge $(x_k)_{k\in\mathbb{N}}$, die rekursiv definiert ist:

$$x_{k+1} = \frac{(x_k + \frac{n}{x_k})}{2}. \tag{2}$$

Das Abbruchkriterium der Iteration ist $x_{k+1} \geq x_k$. Als Initialwert x_0 kann n gewählt werden.

1. Wir implementieren zuerst eine Funktion, welche die Annäherung aus (2) berechnet, indem sie sich rekursiv aufruft, wenn das Abbruchkriterium nicht erfüllt ist:

   ```
   approximate :: Integer→ Integer→ Integer
   ```

   Der erste Parameter ist hier die Zahl n, und der zweite Parameter die momentane Annäherung x_k.

2. Damit implementieren wir die Funktion

   ```
   isqr :: Integer→ Integer
   ```

   welche durch den Aufruf von `approximate` mit dem Initialwert die ganzahlige Wurzel des Arguments berechnet.

3. Um zu prüfen, ob das überhaupt stimmt, schreiben wir jetzt noch eine Funktion

   ```
   testisqr :: Integer→ Bool
   ```

   welche mit Formel (1) prüft, ob die Funktion `isqr` die ganzzahlige Wurzel an dieser Stelle korrekt berechnet. Mit Hilfe dieser Funktion können Sie Ihre Implementation leicht testen.

Die Funktion `isqr` sollte Fehler und Sonderfälle korrekt behandeln.

2 *Im Dienste Ihrer Majestät* *10 Punkte*

In Großbritannien wird Bier in *pints* getrunken, und die Längen in *inches*, *feet*, *yards* und *miles* gemessen. Das verwirrt die meisten Kontinentaleuropäer, daher ist diese Übung unserer dieswöchentlicher Beitrag zur Völkerverständigung.

Für die britischen (die *imperialen*) Maßeinheiten gilt gilt:

- Die kleinste Längeneinheit ist ein *inch*, und entspricht 25.4 Millimetern;
- 12 *inches* sind ein *foot*;
- 3 *feet* sind ein *yard*;
- 1760 *yards* sind eine *mile*.

Definieren Sie zwei Datentypen Metric und Imperial, welche zum einen metrische Längen (in Metern) und zum anderen die imperialen Maßeinheiten modellieren. Damit definieren Sie zwei Funktionen

```
imperialToMetric :: Imperial→ Metric
metricToImperial :: Metric→ Imperial
```

welche zwischen den beiden Maßsystemen konvertieren.

Die Umrechnung sollte beispielsweise 100 Meter in 109 Yards, 1 Fuß, 1 Inch konvertieren. Der Datentyp Imperial sollte das Ergebnis modellieren können; es sollte nicht alles in einen Grunddatentyp eingebettet werden (e.g. alles in Inches oder Millimeter konvertieren). Dabei bleibt es Ihnen überlassen, wie Sie mit den Bruchteilen eines *inch* umgehen.

Um das Ergebnis anzeigen zu können, definieren Sie eine Funktion

```
showImperial :: Imperial→ String
```

welche imperiale Maßeinheiten anzeigt; Null-Werte werden dabei nicht angezeigt. Zum Beispiel ergibt `showImperial(metricToImperial (Meter 100))` die Zeichenkette `"109 yd, 1 ft, 1 in"` (oder vergleichbar).

Formulieren Sie mindestens zwei Eigenschaften, die für die Funktionen gelten müssen, und implementieren Sie diese wie in Aufgabe 1 in einer Funktion mit Wertebereich `Bool`, welche diese Eigenschaften testet. Testen Sie mit diesen Eigenschaften Ihre Implementation.

Zusatzaufgabe: Wer Spaß daran hat, kann noch die heutzutage kaum gebräuchlichen Maßeinheiten *chain*, *fathom* und *furlong* modellieren, die eine Meile unterteilen. Hier gilt:

- 2 *yards* sind ein *fathom*;
- 11 *fathoms* sind ein *chain*;
- 10 *chains* sind ein *furlong*;
- 8 *furlong* sind eine *mile*.