Praktische Informatik 3 WS 2003/04

Berthold Hoffmann <hof>
Christian Maeder <maeder>
Klaus Lüttich <luettich>
Pascal Schmidt <pharao90>
Felix Beckwermert <foetus>
Dennis Walter <dw>
Franck Ngueuleu <nngf>

1. Übungsblatt

Ausgabe: 20.10.03

Bearbeitungszeit: Zwei Wochen

1 *Newton-Verfahren* *14 Punkte*

Isaac Newton (1642 - 1727) erfand nicht nur die Schwerkraft, Trägheits-, Aktions- und Reaktionsgesetze (Kraft ist Masse mal Beschleunigung). Er begründete auch (unabhängig von Leibniz) die Differential- und Integralrechnung.

Das Newton-Verfahren approximiert eine Nullstelle einer differenzierbaren Funktion f als Grenzwert einer Folge (a_i). Dabei ist a_0 ein zu wählender Startwert und a_{i+1} berechnet sich aus a_i durch: $a_{i+1} = a_i - \frac{f(a_i)}{f'(a_i)}$.

Mit Hilfe dieses Verfahrens soll unter Nutzung des Datentyps `Rational` eine beliebig genaue Annäherung der dritten Wurzel aus 2 berechnet werden. (Dazu wählt man als Funktion: $f(x) = x^3 - 2$.)

Die folgenden Funktionen sollen in eine Textdatei `Uebung1.hs` eingetragen werden, die mit folgender ersten Zeile beginnt: '`module Uebung1 where`'. (Modulnamen beginnen mit einem Großbuchstaben.) Zu allen selbstdefinierten Funktionen tragen Sie die Typsignaturen explizit in die Datei ein! (Ansonsten gibt es ggf. kryptische Fehlermeldungen durch Sprachkonzepte, die erst später behandelt werden.) Zeilenkommentaren werden durch '`--`' eingeleitet, Blockkommentare sind von der Form '`{- ... -}`'. Neben dem Code (40 %) fließen die Dokumentation (40%) und die Testfälle (20%) in die Bewertung ein.

Durch '`hugs Uebung1.hs`' oder '`ghci Uebung1.hs`' wird ein Haskell Interpreter aufgerufen, mit dem man Ausdrücke auswerten lassen kann. Innerhalb des Interpreters liefert die Eingabe von '`:?`' Hilfe und '`:q`' beendet die Sitzung. Beachten Sie, dass der Typ `Rational`, im Gegensatz zu `Double` oder `Float`, die rationalen Zahlen exakt repräsentiert. Ebenso verhält es sich mit `Integer` versus `Int`.)

1. Implementieren Sie eine Funktion

   ```
   next :: Rational -> Rational
   ```

   die aus a_i das Folgenglied a_{i+1} berechnet. Die zugehörige Ableitung von f ist $f'(x) = 3x^2$.

2. Geben Sie eine Funktion an

   ```
   nextN :: Int -> Rational -> Rational
   ```

   die zu einem Startwert a_0 das Folgenglied a_i berechnet.

3. Schreiben Sie eine Funktion

```
approx :: Rational -> Rational -> Rational
```

die zu einem sehr kleinen (ersten) Argument ($\varepsilon > 0$) und einem Startwert (zweites Argument) ein Ergebnis liefert, das sich um weniger als ε vom tatsächlichen Wert von $\sqrt[3]{2}$ unterscheidet.

4. Erstellen Sie eine weitere Funktion, die statt des Approximationswertes die Anzahl der nötigen Iterationen liefert. (Alle Funktionsnamen und Variablennamen für formale Funktionsparameter beginnen mit einem Kleinbuchstaben!) Beachten/Prüfen Sie, dass das Ergebnis als Eingabe von `nextN` dasselbe Ergebnis wie `approx` liefert.

5. Erstellen Sie Varianten von den vorherigen Funktionen, die alle ein weiteres Argument (vom Typ `Rational`) erwarten, so dass Sie die dritte Wurzel nicht nur von 2 sondern von beliebigen Zahlen approximieren können.

Folgende, in Haskell vordefinierte, Funktionen könnten für die Lösung nützlich sein:

```
^       :: Rational -> Int -> Rational  -- Infix (Potenzieren)
abs     :: Rational -> Rational
```

$\boxed{2}$ *Schöne Ausgabe* — *6 Punkte*

Jetzt haben wir zwar $\sqrt[3]{2}$ beliebig genau berechnet, aber die Ausgabe der rationalen Zahlen ist nicht sehr befriedigend. Implementieren Sie deshalb eine Funktion

```
showRat :: Int -> Rational -> String
```

die ein rationale Zahl (zweites Argument) als Dezimalzahl mit (maximal) `n` Nachkommastellen ausgeben soll. Beispiel

```
> showRat 10 (1/3)
0.3333333333
> showRat 10 (3/2)
1.5000000000
```

Implementieren Sie ggf. eine Hilfsfunktion, die rekursiv zunächst nur die Nachkommastellen ermittelt. Vordefiniert sind:

```
truncate    :: Rational -> Integer
fromInteger :: Integer -> Rational
++          :: String -> String -> String   -- Infix
show        :: Integer -> String
```

P.S. Arbeit ist Kraft mal Weg (gemessen in Newtonmeter)

Diese Fassung ist vom 20. Oktober 2003.