Praktische Informatik 3 WS 2002/03

1. Übungsblatt

Ausgabe: 21.10.01
Bearbeitungszeit: Zwei Wochen

Christoph Lüth <cxl>
Felix Beckwermert <foetus>
Michael Skibbe <mskibbe>
Dennis Walter <dw>
Rafael Trautmann <pirate>
Thomas Meyer <mclee>

---

1 *Eine Runde Sache* *12 Punkte*

Kreise sind eine runde Sache, und ihr Umfang steht zu ihrem Durchmesser im Verhältnis von ungefähr drei zu eins, oder genauer gesagt π zu eins. Haskell kennt zwar `pi` als vordefinierte Fließkommakonstante, aber in dieser Aufgabe soll eine beliebig genaue Approximation von π errechnet werden. Das ist recht nützlich, wenn man eine große Genauigkeit benötigt, zum Beispiel um die Länger der Erdumlaufbahn auf wenige Zentimeter genau zu bestimmen.

Zur Berechnung von π gibt es viele Näherungsverfahren, aber die meisten sind relativ rechenaufwendig und brauchen endlos und drei Jahre, um vernünftige Näherungen zu produzieren. Hier werden wir den Bailey-Borwein-Plouffe-Algorithmus verwenden, demzufolge π durch folgende Reihe angenähert werden kann:

$$\pi = \sum_{i=0}^{\infty} a_i, \quad \text{mit} \quad a_i = (\frac{1}{16})^i \cdot (\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6}).$$

Unter Nutzung des Datentyps `Rational` soll eine beliebig genaue Annäherung von π implementiert werden.

Implementieren Sie eine Funktion

```
p :: Int-> Rational
```

die π auf die angegebene Anzahl Dezimalstellen berechnet.

Hinweise: Implementieren sie `p`, indem Sie zuerst eine Funktion

```
a :: Int-> Rational
```

implementieren, die das Element a_i der Näherungsreihe berechnet. Damit implementieren Sie eine rekursive Funktion

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

wobei das erste Argument die Genauigkeit ist, das zweite Argument der Index, und das dritte Argument der Näherungswert. `approx` ε k p berechnet das nächste Element p' der Reihe (als $p' = p + a_k$). Wenn die Differenz zwischen p und p' kleiner als die Genauigkeit ε ist, geben wir p zurück, ansonsten wird die nächste Iteration aufgerufen. Die Genauigkeit ε ergibt sich aus der Anzahl der Dezimalstellen d als $\varepsilon = 10^{-d}$.

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

```
^       :: Rational-> Int-> Rational  -- Infix
fromInt :: Int-> Rational
```

Das besondere am Bailey-Borwein-Plouffe-Algorithmus ist übrigens, dass er erlaubt, die n-te Hexadezimalstelle (oder Binärstelle) von π zu berechnen, ohne die Stellen davor zu kennen. Davon machen wir hier allerdings keinen direkten Gebrauch.

$\boxed{2}$ *Schöner Ausgeben* *8 Punkte*

Jetzt haben wir zwar π beliebig genau berechnet, aber die Ausgabe ist nicht sehr befriedigend. Implementieren Sie deshalb eine Funktion

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

wobei `showRat n r` die rationale Zahl `r` in Dezimaldarstellung bis auf `n` Stellen nach dem Komma zurückgeben soll. Beispiel

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

Hinweis: Implementieren Sie eine Hauptfunktion, die den ganzzahligen Teil der Zahl berechnet, und eine Hilfsfunktion aufruft, die rekursiv die Stellen nach dem Komma zurückgibt. Dazu implementieren sie eine Funktion

```
  rest :: Rational-> Rational
```

die den Rest (nach dem Komma) zurückgibt. Folgende, in Haskell vordefinierte Funktionen könnten für die Lösung hilfreich sein:

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

Diese Fassung ist vom 22. Oktober 2002.