Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Lars Noschinski, Dr. Jasmin Blanchette, Dmitriy Traytel

WS 2012/13
22.01.2013
Abgabe: 29.01.2013, 15:30

Einführung in die Informatik 2
13. Übung

Aufgabe G13.1 Dem Zufall auf die Sprünge helfen

Gesucht ist eine Funktion `nRandomR :: (Int, Int) -> Int -> IO [Int]`, die eine Liste von Zufallszahlen generiert. Die Ausgabeliste von `nRandomR (`l`,`h`)` n soll n Elemente enthalten, wobei jedes dieser Elemente $\geq l$ und $\leq h$ sein soll. Außerdem soll die Ausgabeliste keine Elemente doppelt enthalten. Falls in dem vorgegebenen Bereich `(`l`, `h`)` nicht genügend unterschiedliche Zahlen sind, muss Ihre Implementierung nicht terminieren (z.B. für `nRandomR (1,2) 3`).

Verwenden Sie die Funktion `randomRIO :: (Int, Int) -> IO Int` aus `System.Random`, um eine einzige Zufallszahl aus dem vorgegebenen Bereich zu erzeugen. Duplikate vermeiden Sie mit einem Hilfsargument für bereits "gewürfelte" Zahlen. So erkennen Sie Wiederholungen und können einfach neu würfeln.

Aufgabe G13.2 Zahlenraten

Wir wollen folgendes Spiel in Haskell implementieren: Zunächst wählt das Programm eine zufällige Zahl zwischen 0 und 100 (einschließlich) und fragt dann den Benutzer nach einer Zahl. Hat der Benutzer die richtige Zahl erraten, gibt das Programm die Anzahl der Versuche zurück. Andernfalls zeigt es an, ob die gewählte Zahl größer oder kleiner ist und fragt erneut nach einer Eingabe, bis der Benutzer die richtige Zahl erraten hat.

1. Schreiben Sie zunächst eine IO-Aktion `getLineInt :: IO Int`, die eine Zeile von der Tastatur einliest. Repräsentiert diese Zeile eine (positive) Zahl, so soll die Zahl zurückgegeben werden, andernfalls soll eine Fehlermeldung ausgegeben werden und erneut eine Eingabe gelesen werden. Dies wiederholt sich solange, bis es eine gültige Eingabe gibt.

   Verwenden Sie die IO-Aktion `getLine :: IO String` um eine Zeile zu lesen. Mit der Funktion `read :: String -> Int` können Sie eine gültige Eingabe in eine Zahl umwandeln.

2. Verwenden Sie die vorherige Teilaufgabe um das Spiel als IO-Aktion `guessNum :: IO Int` zu implementieren.

Hinweis: Die Aufgaben zum Thema IO sind leicht manuell und schwer automatisch zu testen. Deswegen gibt es zu diesem Übungsblatt keine QuickCheck-Tests. Testen Sie Ihr Programm vor der Abgabe!

Aufgabe H13.1 Interaktiv Sortieren (10 Punkte)

Schreiben Sie eine IO-Aktion `getSorted :: IO [String]`, die zeilenweise solange Text einliest, bis eine leere Zeile eingegeben wurde. Geben Sie dann die Liste der eingegeben Zeilen sortiert zurück. Benutzen Sie als einzige vordefinierte IO-Aktion `getLine :: IO String`.

Beispiel: Wird `getSorted` aufgerufen und gibt der Benutzer die folgenden Zeilen, gefolgt von einer Leerzeile, ein,

```
how
are
you
my
friend
```

so soll das Ergebnis `["are","friend","how","my","you"]` sein.

Aufgabe H13.2 Von Bullen und Bären (10 Punkte)

Wir wollen folgendes Spiel in Haskell implementieren: Zunächst wählt das Programm eine vierstellige zufällige Zahl mit unterschiedliche Ziffern (0 als erste Ziffer ist erlaubt) und hält diese geheim. Der Benutzer hat zehn Versuche um die Zahl zu erraten. Dabei gibt er bei jedem Versuch ebenfalls eine vierstellige Zahl mit unterschiedlichen Ziffern ein. Der Computer antwortet mit zwei Zahlen: (1) Mit der Anzahl der brav im Stall stehenden Bullen (d.h. Ziffern, die in der geheimen Zahl an der gleichen Stelle stehen wie in der Benutzereingabe) und (2) mit der Anzahl der wild herumstreunenden Bären (d.h. Ziffern der Benutzereingabe, die zwar in der geheimen Zahl vorkommen, aber nicht an der gleichen Stelle stehen).

Zum Beispiel: falls die geheime Zahl 1234 und die Benutzereingabe 9213 ist, so sollte die Antwort des Programms 12 sein: 1 für die Ziffern an den richtigen Stellen (hier nur die 2) und 2 für die Ziffern an den falschen Stellen (hier 1 und 3).

Nach Ende des Spiels soll die Ausgabe insgesamt wie nachfolgend dargestellt aussehen. Dabei ist in beiden Fällen (a) und (b) lediglich der mittlere Block vom Spieler eingegeben. Links davon werden die Versuche gezählt und rechts stehen die Antworten des Programms auf die jeweilige Eingabe. Definieren Sie das Spiel als die IO-Aktion `bullsAndBears :: IO ()`.

Hinweise:

1. Verwenden Sie `nRandomR :: (Int, Int) -> Int -> IO [Int]` aus G13.1 um die Ziffern der Geheimzahl zu erzeugen. Generell ist es leichter mit einer Liste von vier Ziffern zu arbeiten, anstatt mit der vierstelligen Zahl.

```
   ****
 1 0123 10
 2 4567 01
 3 0894 02
 4 8925 02
 5 6198 03
 6 9683 40
You won!
```

(a) Sieg des Spielers

```
   ****
 1 1234 01
 2 2345 00
 3 3456 10
 4 4567 02
 5 5678 11
 6 6789 02
 7 7890 02
 8 8901 02
 9 9012 02
10 0123 20
You lost! Secret number was 0176
```

(b) Niederlage des Spielers

2. Verwenden Sie `putStr :: String -> IO ()` und `putStrLn :: String -> IO ()` für die Ausgabe. Die Funktion `show :: Show a => a -> String` kann zur Umwandlung von Zahlen in Strings genutzt werden.
3. Verwenden Sie `getChar :: IO Char` um Benutzereingaben zeichenweise einzulesen und `digitToInt :: Char -> Int` aus `Data.Char` um diese in Ziffern umzuwandeln.
4. Gibt der Benutzer ein ungültiges Zeichen ein (z.B. Buchstaben oder Ziffern, die der Benutzer im laufenden Versuch bereits eingegeben hat), so soll das Zeichen einfach ignoriert werden.
5. Strukturieren Sie Ihr Programm! Verschiedene Hilfsfunktionen für die Benutzereingabe, das Berechnen der Antwort und das Spiel selbst ausgestattet mit einem Versuchszähler (als Parameter) sind Pflicht.

Aufgabe W13.1 Für die van Goghs dieser Welt (**0 Punkte**, letzte Wettbewerbsaufgabe)

In der Vorlesung und vorherigen Aufgabenblatt wurde das Thema ASCII-Art reichlich besprochen. In dieser Aufgabe geht es darum, ein Pixel- oder Vektor-Bild in einem Standardformat (z.B. PNG, SVG, PDF) zu generieren. Alle Bibliotheken von Drittanbietern sind ausdrücklich erlaubt. Siehe zum Beispiel:

`http://www.cs.yale.edu/homes/hudak/SOE/`
`http://www.haskell.org/haskellwiki/Applications_and_libraries/Graphics`

Die Bewertungskriterien sind Ästhetik („hübsches Bild!“) und Technik („toller Code!“). Der Jury besteht aus dem Meta-Master, dem Tutoren-Master und den (Co-)Masters of Competition.

Diese Wettbewerbsaufgabe ist getrennt in einer Datei namens `Gogh.hs` einzureichen. Zusammen mit `Gogh.hs` sollten sie auch das generierte Bild einreichen unter den Namen `gogh.`*xxx*, wo *xxx* z.B. `png` ist.

Wichtig: Wenn Sie diese Aufgabe als Wettbewerbsaufgabe abgeben, stimmen Sie zu, dass Ihr Name ggf. auf der Ergebnisliste auf unserer Internetseite veröffentlicht wird. Sie können diese Einwilligung jederzeit widerrufen, indem Sie eine Email an `fp@fp.in.tum.de` schicken. Wenn Sie nicht am Wettbewerb teilnehmen, sondern die Aufgabe allein im Rahmen der Hausaufgabe abgeben möchten, lassen Sie bitte die {`-WETT-`} . . . {`-TTEW-`} Kommentare weg. Bei der Bewertung Ihrer Hausaufgabe entsteht Ihnen hierdurch kein Nachteil.