Lehrerbildungsforum Informatik

Einführung in die Programmierung und das Problemlösen in Haskell
29. September, Frankfurt am Main

Prof. Dr. Manfred Schmidt-Schauß und PD Dr. David Sabel

Vorschlag für eine Haskell Programmieraufgabe für das Forum mit dem Thema: programmierte Textverarbeitung

Für diejenigen, die schon vorher Haskell ausprobieren wollen und evtl. schon versuchen zu programmieren.
Es wird im Forum aber auch eine verdichtete Basiseinführung geben, so dass es nicht nötig ist, diese Aufgaben zu bearbeiten. Die Aufgaben selbst und Teile der Lösung werden am Ende des Forums besprochen.

Eine Serie von Beispielaufgaben wird bereitgestellt, die von den Teilnehmern schon vorab versuchsweise angegangen werden kann. Zielgruppe sind diejenigen, die schon Zeit und Lust haben, sich mit Haskell und Haskell-Programmierung zu beschäftigen.
Konkretes Ziel *ist es, dass die interessierten Teilnehmer das* `main` *Programm lauffähig haben im* `ghci`*-Interpreter und schon einige Teilaufgaben versucht haben zu Programmieren und zu Testen*
Weitere Ziele*: Durch Programmieren und Testen sollen die interessierten Teilnehmer selbst möglichst viele Aspekte sehen und erkennen:*
Datenstrukturen, Rekursion, Aufteilung der Aufgaben auf Funktionen, vorgegebene Funktionen/Bibliotheken, Typen, Typfehler, Interpreter-Umgebung des `ghci`*; Schwierigkeitsgrade; usw.*

Am 29.9 würden wir dann innerhalb der Zeit direkt Fragen zur Programmierung, Compilierung, Testen und Korrektur der Programme bzw. Funktionen klären und diskutieren.

Aufgabe zur programmierten Textverarbeitung

Die Aufgabe bzw. die Teilaufgaben bestehen darin, einen vorgegeben Text im Textfile „`eingabe.txt`“ einzulesen, mit einer selbst geschriebenen Haskell-Funktion zu manipulieren und das Ergebnis in einen file „`ausgabe.txt`“ zu schreiben.
Ein Vorteil der Vorgehensweise ist, dass man das Ergebnis dann in einem Texteditor sofort sehen kann.

Da man die imperative Schnittstelle von Haskell in der Schule eher nicht zum Programmieren verwenden sollte, ist diese bereits im Programmfile „`text-verarbeitung.hs`“ vorgegeben. Die Verarbeitungs-Funktionen sind als Lücken gegeben.
Es sind auch weitere Vorschläge zur Aufteilung der Funktionen auf Unterfunktionen und Typen usw. enthalten.

Zur Programmierung sollten möglichst die Standardfunktionen der Listenverarbeitung aus dem Standard-Prelude und aus der Bibliothek `Data.List` (oder weiteren wie `Data.Char`) verwendet werden. Der File soll nach dem Einlesen direkt als eine Liste von Zeilen dargestellt werden, wobei eine Zeile selbst wieder eine Liste von Strings ist. Leerzeichen (Blanks) werden als Trennzeichen verwendet.

Um die Aufgaben zu lösen, ist es sehr sinnvoll, Unterfunktionen zu schreiben, die in mehreren Fällen rekursiv zu programmieren sind. Die Hauptfunktionen sollten relativ kurz sein und die Unterfunktionen verwenden,

Die unabhängigen Einzelaufgaben:

1. Lösche alle Worte der Länge > 4 (als Beispiel) .
2. Erzeuge ein Ausgabe-File, bei dem jedes Wort in einer eigenen Zeile steht.
3. Umkodieren: Ersetze alle Umlaute durch deren LaTeX-Kodierung:
   ä durch `\"a` usw. und ß durch `{\ss}`.
4. Im Eingabefile sollen alle Zahlen $1, \ldots, 12$ durch Zahlworte ersetzt werden.
5. Breche Zeilen um, wenn diese länger sind als eine Standardlänge, z.B. 80. Leichte Verbesserung der Funktion: Worte sollen nicht durchgeschnitten werden, sondern komplett übertragen werden, außer wenn es nicht anders geht, weil das Wort selbst länger als die erlaubte Zeilenlänge ist.
6. Gebe alle Zeilen rechtsbündig statt linksbündig aus.
7. Gebe alle Zeilen im Blocksatz aus (annähernd). Verwende dazu die Unterfunktion `verteileMN`

Vorschläge für Hilfsfunktionen / Konstanten.

1. `anzahlWorte:: [String] -> Integer` Berechnet Anzahl der Worte in einer Zeile.
2. `anzahlZeilen:: [a] -> Integer` Berechnet Anzahl der Zeilen.
3. `zeilenLaenge:: [String] -> Integer` Berechnet Länge einer Zeile in der gedruckten Form.

4. `standardLaenge = 80`

5. `verteileMN m n`: Die Eingaben m, n sollen ganzzahlig und nicht negativ sein, und $n > 0$ sein. Die die Anzahl m der Blanks ist zu verteilen auf die Anzahl n der Lücken zwischen den Worten (einer Zeile); z.B. 13 auf 5 verteilen kann als Ergebnis haben: $[3, 3, 3, 2, 2]$.