Einführung in Haskell

Haskell ist eine moderne, funktionale Programmiersprache.

1. Motivation zum Einsatz einer funktionalen Programmiersprache bzw. eines deklarativen Programmierparadigma

- Während in **imperativen** Programmiersprachen das Konzept der Variablen als Speicherbereich für Werte im Vordergrund steht, ist in funktionalen Programmiersprachen der Begriff der **Funktion** von zentraler Bedeutung.
- Die zentrale Anweisung in imperativen Programmiersprachen ist die Zuweisung. Ein Problem wird durch die schrittweise Veränderung der Werte der Variablen gelöst. Dabei spielt die Reihenfolge der Anweisungen eine entscheidende Rolle. Diese kann durch Kontrollstrukturen (wie z.B. Verzweigungen oder Wiederholungen mit Hilfe von Schleifen) beeinflusst werden.
- In **funktionalen Programmiersprachen** wie Haskell werden Berechnungen hingegen vornehmlich durch die **Auswertung von Ausdrücken** durchgeführt.
  Ein funktionales Programm ist eine Menge von (Funktions-)Definitionen.
  Die Ausführung wird in funktionalen Programmiersprachen durch die Auswertung von Ausdrücken (in ihrer Umgebung) durchgeführt.

2. Funktionen mit Haskell

Ein Skript (Programm, Haskell-Datei) besteht aus einer Sammlung von **Funktionen**. Dabei gelten für Haskell folgende **Regeln**:

- Der Dateiname (Skriptname) darf **keine Leerzeichen** oder **Umlaute** enthalten
- Die Reihenfolge der Funktionen spielt keine Rolle.
- Funktions**namen müssen** mit einem Kleinbuchstaben beginnen.
- Funktions**namen** dürfen keine Minuszeichen enthalten.
- An einigen Stellen muss auf eine korrekte Einrückung geachtet werden.
- Kommentare werden durch zwei Minuszeichen (--) eingeleitet.
- Groß- und Kleinschreibung ist zu beachten und **wird unterschieden.**

3. Aufbau einer Funktion

Ähnlich zur Mathematik[1] definiert man in Haskell den linken Teil einer Funktion durch den rechts des Gleichheitszeichens stehenden **Ausdruck**. Auf der linken Seite steht zuerst der Name der Funktion gefolgt von Parametern, die durch Leerzeichen getrennt werden.

```
-- Quadrat einer Zahl
-- Datei: mathematik.hs
quadrat :: Int -> Int -- Typdefinition; kann entfallen
quadrat x = x*x -- Funktionsdefinition
```

4. Umgang mit Programmdateien in WinHugs, dem Haskell-Interpreter

''Programme'' bestehen in Haskell aus Funktionen, die in einer Textdatei mit der Endung .hs abgespeichert sind. Diese Datei kann in WinHugs mit einer Eingabe am **Prompt** (die Eingabaufforderung im WinHugs-Interpreter)
`:l` datei.hs (load)
geladen werden. Anschließend stehen alle darin definierten Funktionen zur Verfügung.
Sobald eine Datei geladen wurde, kann diese mit :e bearbeitet werden. Das Speichern im Editor veranlasst WinHugs zum Neueinlesen. Es bietet sich allerdings an, einen **eigenen Editor** zum Bearbeiten zu nutzen!
—> Mögliche Fehler werden allerdings nur angezeigt, wenn mit **:r (reload)** ein Neueinlesen erzwungen wurde!
Hier eine unvollständige Liste von (Win-)HUGS Kommandos. Diese gelten auch für den GHCI (Haskell-Interpreter):

`:l` datei.hs (load)
`:r` (reload)
`:b` (browse module)
`:e` (edit)
`:?` (help)

5. Zusammenfassung

- Funktionen werden in Skriptdateien definiert (Datei-Endung .hs).
- Selbstdefinierte Funktionen müssen mit einem Kleinbuchstaben beginnen.
- Am Prompt von WinHugs werden die Funktionen mit konkreten Werten aufgerufen.
- Mit *:t (type)* kann der Typ einer Funktion abgefragt werden.
- Mit *:b (browse)* werden alle Funktionen der Programmdatei angezeigt.
- Typ-Signaturen müssen nicht (können aber) angegeben werden. Beispiel: `quadrat :: Int -> Int`

[1] z.B. $y = x + 2x$ oder $f(x) = 3x^3$, aber auch $y = 15$ oder $y = 54 - 23$ oder $meinepotenz(x) = 13^{12}$ ist möglich.