technische universität dortmund

FLDIT

Lehrstuhl 1
Logik in der Informatik

Prof. Dr. Peter Padawitz

Jos Kusiek (*jos.kusiek@tu-dortmund.de*) Wintersemester 2016/2017

Übungen zu Funktionaler Programmierung
Präsenzblatt 1

Aufgabe 1.1

Installieren Sie die Haskell-Platform (`http://www.haskell.org/platform/`) auf ihrem Rechner. Stellen Sie dabei sicher, dass `ghc` und `ghci` zu ihrer Pfadvariablen hinzugefügt sind.

1. Öffnen Sie den Texteditor Ihrer Wahl und tippen Sie folgendes Programm ab:

```
f :: Int -> Int -> Int -> Int
f x y z = x + y + z * z
```

2. Speichern Sie das Programm in einer Datei mit der Endung `.hs`. Den Pfad zu der Datei nennen wir im Folgenden `file.hs`.

3. Öffnen Sie die Kommandozeile und laden Sie die Datei mit dem interaktiven Modus des GHC (GHCi genannt), wie folgt: `ghci file.hs`
   Sie sollten nun die folgende Ausgabe erhalten:

```
[1 of 1] Compiling Main             ( file.hs, interpreted )
Ok, modules loaded: Main.
*Main>
```

4. Rufen Sie nun die Funktion `f` auf, indem Sie zum Beispiel `f 1 2 3` eingeben und mit ENTER bestätigen. Das Ergebnis wird ausgegeben und Sie können weitere Funktionsaufrufe auswerten lassen.

Folgende Kommandos des GHCi haben sich als nützlich erwiesen:

- `:load file` (kurz `:l`) lädt die Datei `file` in den GHCi.
- `:reload` (kurz `:r`) lädt die aktuelle Datei neu ein. Nachdem Änderungen an dem Quelltext vorgenommen wurden, kann die aktuelle Datei mit `:r` leicht neu geladen werden.
- `:type ausdruck` (kurz `:t`) zeigt den Typ des Ausdruckes `ausdruck` an, z.B. `:t f` oder `:t f 1 2 3`.
- `:help` (kurz `:h`) öffnet die Hilfe mit weiteren nützlichen Befehlen.
- `:quit` (kurz `:q`) beendet den GHCi.

Aufgabe 1.2

Die folgende Aufgabe enthält eine Reihe von fehlerhaften Haskell-Ausdrücken. Ziel dieser Aufgabe ist, dass Sie sich mit den Fehlermeldungen des GHCi vertraut machen. Laden Sie dazu die Datei aus Aufgabe 1 und interpretieren Sie die folgenden Ausdrücke mit dem GHCi. Versuchen Sie die Fehlermeldungen nachzuvollziehen.

1. `f 3 1 True`

2. `f 4 3 2 1`
3. `f 3 2 1`
4. `foo 3 2 1`

Aufgabe 1.3

Gegeben sei die Funktion `addFive` und die konstante Funktion `one`:

```
addFive :: Int -> Int
addFive x = x + 5

one :: Int
one = 1
```

1. Definieren Sie eine Konstante `k :: Int`, deren Auswertung die Zahl `11` liefert. Nutzen Sie dabei nur die Funktionen `addFive` und `one`.
2. Definieren Sie eine Funktion `addTen :: Int -> Int`, die eine ganze Zahl als Parameter erhält und die Summe aus Zehn und der übergebenen Zahl berechnet. Nutzen Sie dafür die Funktion `addFive` und den Dollar-Operator.
3. Definieren Sie eine Funktion `addTenComposition :: Int -> Int`, die semantisch äquivalent zu der Funktion `addTen` ist, jedoch die Funktion `addFive` und die Funktionskomposition nutzt.

Aufgabe 1.4

Werten Sie den folgenden Ausdruck unter Angabe von Zwischenergebnissen aus:

$$(\lambda y.(\lambda x.x + 1)\ (4 + y))\ 6$$