Prof. Dr. R. Loogen, M. Dieterle
Fachbereich Mathematik und Informatik
Hans-Meerwein-Straße
D-35032 Marburg

14. Oktober 2009

Philipps Universität Marburg

Übungen zu „Grundlagen des Compilerbau“, Winter 2009/10

Nr. 1, Abgabe der Aufgaben: 21. Oktober 2009 vor der Vorlesung

---

Hinweise:

- Die Lösungen aller Aufgaben sollen grundsätzlich schriftlich abgegeben werden. Programme sollen ausgedruckt und *zusätzlich* per E-Mail an Ihren Tutor geschickt werden. Die Abgabe ist in Gruppen bis zu zwei Personen erlaubt.
- Ihre Programme müssen kompilierbar sein, um bepunktet zu werden.
- Die Präsentation von Aufgaben im Tutorium ist Bedingung für den Erwerb eines Leistungsnachweises. Bitte achten Sie selbst (mit) darauf, dass Sie diese Bedingung erfüllen können.
- Wir arbeiten mit der Programmiersprache Haskell. Einen kompakten Überblick über Haskell-Funktionen und Datentypen finden Sie unter:
  http://zvon.org/other/haskell/Outputglobal/index.html .
  Auch die Haskell Suchmaschine Hoogλe ist eventuell nützlich:
  http://haskell.org/hoogle .

Aufgaben

1.1 **T-Diagramme** — 4 Punkte

In GNU-Compilern wird eine plattformunabhängige Maschinensprache namens RTL als Zwischensprache verwendet. Es gibt Übersetzer, die in Hochsprachen wie **C**, C++ oder Pascal geschriebene Programme nach RTL übersetzen; analog gibt es Übersetzer, die RTL-Programme auf Plattformen wie z.B. x86 oder PPC spezialisieren. Außerdem gibt es einen RTL-Optimierer, der ein RTL-Programm in ein effizienteres RTL-Programm transformiert. Alle Übersetzer sind in C implementiert.

(a) Zeigen Sie mit Hilfe von T-Diagrammen, wie Sie diese Übersetzer und den Optimierer mit einem x86-Rechner lauffähig machen können, wenn Sie auf dem Rechner über einen lauffähigen C-Compiler verfügen.

Zeigen Sie mit Hilfe von T-Diagrammen, wie Sie auf einem x86-Rechner mit lauffähigem C-Compiler

(b) einen **optimierten** Pascal nach x86-Maschinensprache Compiler erzeugen, der auf einer x86-Maschine läuft.

(c) einen C++ nach PPC-Maschinensprache Compiler erzeugen, der auf einer PPC-Maschine läuft.

Bitte wenden!

1.2 Entfernung von Whitespace

3 Punkte

Schreiben Sie ein Haskell-Programm, das aus einer gegebenen Zeichenkette Reihen von Leerzeichen, Tabulatoren (`\t`) und Zeilenumbrüchen (`\n`) entfernt und durch jeweils ein Leerzeichen ersetzt.

(a) Schreiben Sie zunächst eine Funktion `isWhiteSpace :: Char → Bool` die erkennt, ob der Argumentbuchstabe zu den oben genannten Zeichen gehört.

(b) Schreiben Sie nun eine Funktion `compress :: [Char] → [Char]` die unter Benutzung von `isWhiteSpace` Ketten von mehr als einem Leerzeichen zu einem einzigen Leerzeichen reduziert.

Beispiel:

```
"Emil und   die Detektive\n\n \t\t von Erich Kaestner    "
```

wird konvertiert zu

```
"Emil und die Detektive von Erich Kaestner "
```

1.3 Scannen eines Strings

5 Punkte

(a) Definieren Sie eine Haskell-Funktion `getStrings :: String → [String]`, die einen `String` erhält und alle zusammenhängenden Zeichenketten extrahiert, welche nur aus Kleinbuchstaben, Großbuchstaben, Zahlen, Leerzeichen und dem '_' - Symbol bestehen. Sie können dazu die Funktionen
`isLower, isUpper, isDigit, isSpace :: Char → Bool` benutzen.

(b) Schreiben Sie ein Haskell-Programm, das die Funktion `getStrings` auf die kompilierte Version des eigenen Codes anwendet. Die von `getStrings` erstellte Wortliste soll ausgegeben werden. Kompilieren Sie das Programm zu Testzwecken mit "ghc". Nützliche Funktionen sind:

```
getProgName :: IO String
readFile    :: String → IO String

putStrings :: [String] → IO [()]
putStrings = mapM putStrLn
```