



# Universität Karlsruhe (TH)

## Lehrstuhl für Programmierparadigmen

Compilerpraktikum SS 2010

Dozent: Prof. Dr.-Ing. G. Snelting

Betreuer: Matthias Braun

Betreuer: Sebastian Buchwald

<http://pp.info.uni-karlsruhe.de/>

[snelting@ipd.info.uni-karlsruhe.de](mailto:snelting@ipd.info.uni-karlsruhe.de)

[matthias.braun@kit.edu](mailto:matthias.braun@kit.edu)

[sebastian.buchwald@kit.edu](mailto:sebastian.buchwald@kit.edu)

Übungsblatt 2

Ausgabe: 21.4.2010

Besprechung: 28.04.2010

### Aufgabe 1: Lexikalische Analyse

Die lexikalische Analyse zerteilt die Eingabe in die grundlegenden Symbole (auch Token genannt) der Sprache. Die lexikalische Analyse ist strenggenommen nur eine Optimierung, die die Eingabe in eine Form bringt die besser für die syntaktische Analyse geeignet ist. Sie führt zu einer Informationskompression da längere Zeichenketten für Bezeichner und Schlüsselwörter auf Symbole abgebildet werden. Da die Techniken zur lexikalischen Analyse (endliche Automaten) sehr schnell arbeiten, lohnt sie sich im allgemeinen.

#### 1.1 Planung

- Lesen Sie sich den Teil des Sprachberichts zur lexikalischen Analyse durch.
- Entwerfen Sie ein Interface für ihren Lexer. Wie sehen die Datenstrukturen für die Tokens aus?
- Welche Rolle hat die Stringtabelle?
- Entwerfen Sie einen deterministischen endlichen Automaten für die Tokens aus ihrer Sprache.

#### 1.2 Implementierung

- Implementieren Sie ihren Lexer nach einem auf den Folien angedeuteten Verfahren.

#### 1.3 Testen

Zum Testen der lexikalischen Analyse bietet es sich an einen gegebenen Quelltext einzulesen und die erkannten Tokens nacheinander auszugeben. Ihr Programm sollte sich dabei an folgende Standards halten:

Ihr Compiler sollte bei Aufruf mit `compiler --lextest dateiname` die Eingabe aus der angegebenen Datei einlesen und dann folgende Ausgabe produzieren:

- Für Leerräume und Kommentare wird nichts ausgegeben.
- Schlüsselwörter und Operatoren werden so wie sie sind ausgegeben.
- Bezeichner werden mit dem Präfix `identifier` ausgegeben.
- Zahlen werden mit dem Präfix `integer literal` ausgegeben.
- Nach jedem ausgegebenen Token wird eine neue Zeile begonnen.
- Tritt ein Fehler auf, so wird eine Meldung ausgegeben die die Zeichenkette `error` enthalten muss. Ob man einem Fehler abgebrochen oder weiter analysiert wird ist nicht spezifiziert.
- Ist das Ende der Eingabe erreicht so wird `EOF` ausgegeben.

Ihr Compiler sollte im `--lextest` Modus diese Vorgaben strikt einhalten da wir automatisierte Tests damit durchführen wollen.

Beispiel: Die Eingabe:

```

/**
 * A classic class
 * @author Beate Best
 */
public class classic {
    public int method(int arg) {
        int res = arg+42;
        res >>= 4;
        return res;
    }
}

```

Sollte die folgende Ausgabe produzieren:

```

public
class
identifier classic
{
public
int
identifier method
(
int
identifier arg
)
{
int
identifier res
=
identifier arg
+
integer literal 42
;
identifier res
>>=
integer literal 4
;
return
identifier res
;
}
}
EOF

```