Lexikalische Analyse

Um den Entwurf, bzw. die Entwicklung eines Compilers einfacher - oder besser gesagt übersichtlicher - zu gestalten, lässt sich durch die Aufteilung der Lexikalischen und syntaktischen Analyse die Codecompilierung nach dem Teile und Herrsche Prinzip in kleinere Aufgaben herunterbrechen. Müsste der Parser zusätzlich zur Syntaxüberprüfung noch nach z.B. Whitespaces (also Leerzeichen, etc.) ausschau halten, so wäre der entstehende Compilercode deutlich umfangreicher. Insbesondere bei der Entwicklung einer neuen Sprache führt die Aufteilung des Lexens und des Parsens zu einem insgesamt saubereren Design. Auch die Effizienz des entstehenden Compilers wird gefördert, da ein separates Lexen die Möglichkeit bietet spezialisierte Techniken zu verwenden, welche dass Parsen allerdings nicht unterstützen würden. Weiterhin existieren spezialisierte Buffer-Technologien für das Einlesen von Zeichenketten Darüber hinaus wird die Kompatibilität des Compilers erhöht, da Geräte-spezifische Eigenheiten auf die Überprüfung des Lexikons beschränkt werden kann

Im Wesentlichen interessieren wir uns in dieser Phase für drei ähnliche, aber klar definierte, voneinander abgegrenzte Begriffe. Ein Token ist is ein Name-Wert Paar, wobei der Wert hierbei optional ist und der Name abstrakt die Art der lexikalischen Einheit beschreibt (z.B. ein Buchstabe, eine Zahl, ...). Ein Pattern (deutsch *Muster*) beschreibt in welcher Reihenfolge die Zeichen eines Lexems des Tokens angeordnet sein dürfen, wobei ein Lexem nun eine Instanz des Tokens innerhalb des Quelltextes handelt. Der Lexer eines Compilers für PHP würde in dem Befehl

```
<?php
    echo "Hello World!";
?>
```

“<?php”, “echo” und “?>” als Lexeme feststellen, welche dem Muster eines Token des **id**-Typen entsprechen und “Hello World!” als Lexems eines **literal**. Natürlich existieren neben den bereits erwähnten Tokentypen noch einige zusätzliche, welche in der untenstehenden Tabelle [tab:compiler_lex_token_examples] beschrieben sind.

l|l|l Token & Abstrahierte Beschreibung & Beispiel if & Buchstaben i, f & if else & Buchstaben e, l, s, e & else comparison & or , = or =, == or != & =, != id & Buchstabe, auf welchen Buchstaben oder Zahlen folgen & pi, x2, tmp_var number & Jegliche numerischen Konstanten & 3.1415926535, 0, 2.2e22 literal & Alles von einem paar " eingeschlossene & “Hello, World!”