4. Die lexikalische Analyse

Beispiele für reg. Ausdrücke:

- Haus[a-z]* deckt alle Strings ab, die mit dem Teilwort Haus beginnen
- [QY][a-z]* deckt alle Wörter ab, die mit Q oder mit Y beginnen
- ^…$ deckt einen bel. 3-Zeichen-String ab, wenn er allein in einer Zeile steht
- [^a-z] deckt ein Zeichen ab, das kein Kleinbuchstabe ist
- ein|eine deckt die Strings ein und eine ab
- ein[e]? wie oben
- (abc)* deckt den leeren String, abc, abcabc, abcabcabc, … ab
- [aeiouAEIOU]{2,2} deckt alle Doppelvokale ab

4. Die lexikalische Analyse

Aufgabe: Schreiben Sie einen regulären Ausdruck der:

- alle Wörter (Anfänge) im Text entdeckt, die wegen Zeilenende getrennt werden.
- Alle Vorkommen Ihres Nachnamens im Text erkennt.
- Alle Vorkommen Ihres Nachnamens, wobei zuvor der Anfangsbuchstabe eines bel. Mitglieds Ihrer Familie stehen soll

  (Beispiel: F. Meyer oder P.Meyer oder S. Meyer, mit/ohne Blank zwischen . und Nachnamen)

Aufgabe: Schreiben Sie eine Regelmenge, die Bezeichner erkennt, dabei aber zwischen Arraynamen (a[..]), Funktionsnamen (f()) und einfachen Bezeichnern unterscheidet.

4. Die lexikalische Analyse

lex-Regeln: Wissenswertes

- | anstelle einer Aktion: hier gültige Aktion steht bei nächstem Pattern
- yytext ist char-Array mit durch Pattern abgedecktem Eingabestring
- ECHO Kurzform für printf(“%s“, yytext);
- yylineno Variable, die pro Eingabezeile akt. Zeilennummer enthält
- yyleng Variable mit Länge des Strings in yytext
- yyless(n) läßt nur die ersten n Zeichen von yytext in yytext, schiebt die restl. in Eingabe zurück (Lookahead-Möglichkeit)
- input(), output(c), unput(c): vordefinierte lex-Makros,
- input() liest nächstes Zeichen aus Eingabetext,
- output(c) schreibt c auf Standardausgabe
- unput(c) schiebt Zeichen c in Eingabetext zurück
- REJECT Makro, nur als letztes einer Aktion, stellt String in yytext für weitere Regeln zur Verfügung

4. Die lexikalische Analyse

Einsatzbereiche von lex:

- für beliebige Umformungen/Analysen von Text
- speziell mit Parsern (yacc), wo Eingabetext bestimmte Grammatik besitzt
- Sonst. Anwendungen, z.B.
  - Erstellung Cross-Referenz-Liste für Module
  - Taschenrechner (in Postfix-Notation)
  - Klammerungs-Analyse
  - Komplexitätsanalysen von Shell-Skripten
  - usw.

4. Die lexikalische Analyse

Jflex:

- geringfügig anderer Aufbau:

```
Benutzercode
%%
Optionen/Deklarationen/Makros/Zustandsnamen
%%
Lex-Regeln
```

- Benutzercode wird unverändert in lexer-Klasse übernommen (vor class-Deklaration)
- Optionen: für Klassenname, Zeichensatz, Zeilen-/Spaltenzähler,…
- Deklarationen: zwischen %{ ….%} für benötigte Variablen, Funktionen
- Makros: `Zeilentrenner = \r|\n|\r\n`

4. Die lexikalische Analyse

Fragestellungen zu Kapitel 4:

- Aufgabe der lexikalischen Analyse?
- Was macht der Sieber? Was erzeugt der Scanner?
- Womit lassen sich Tokenklassen beschreiben?
- Wie ist ein endlicher Automat definiert?
- Syntax/Semantik regulärer Ausdrücke
- Bestandteile einer flex-Datei
- Syntax/Semantik der lex-Regeln
- Erstellen und Übersetzen einer flex-Datei am Beispiel
- Was macht yywrap? Was enthält yytext?