Universität Ulm - Abteilung Angewandte Informationsverarbeitung
Prof. Dr. Franz Schweiggert und Hans Braxmeier, Wintersemester 2002

6. Übungsblatt zur Vorlesung Allgemeine Informatik I

Abgabetermin: Mittwoch, 04.12.2002

Aufgabe 1: Reguläre Ausdrücke *(7 Punkte)*

In den vorhergehenden Übungsblättern wurden Aufgaben zu Formalen Sprachen, Grammatiken und endlichen Automaten gestellt. Wozu das Ganze werden sich einige von Euch gefragt haben. Nun, Formale Sprachen, Grammatiken und endliche Automaten sind nicht nur graue Theorie, sondern finden vielfältige Verwendung in der Praxis. Z.B. sind reguläre Grammtiken und endliche Automaten ein äquivalentes Beschreibunggsystem für reguläre Ausdrücke! Toll!? Und was sind nun reguläre Ausdrücke? Wozu braucht man die?

Ein regulärer Ausdruck ist - stark vereinfacht gesagt - eine Zeichenfolge, die ein Textmuster repräsentiert. Dieses Textmuster kann dann in einer längeren Zeichenkette gesucht werden. Ein umgangssprachlich formuliertes Textmuster ist etwa: ein Wort, das mit J beginnt und zwei a's enthält. Auf dieses Textmuster würden beispielsweise sowohl Januar als auch Java passen. Der Einsatz ergibt sich dann beispielsweise bei der Wortsuche auf Internetseiten, die sehr viel Text enthalten.

Super und dann muß ich jedes Mal die Seite downloaden und mit egrep meine Datei durchsuchen??? Nein!!! Java (eine Programmiersprache), der Internetexplorer, ja und man höre und staune, selbst Word kann reguläre Ausdrücke verarbeiten. Also so abgespacet ist die graue Theorie dann doch wieder nicht...

So jetzt aberschnell zu den Aufgaben: Um den Umgang mit regulären Ausdrücken zu lernen, ist in der folgenden Aufgabe die Datei woerter.txt gegeben, in der Ihr nach bestimmten Textmustern suchen sollt (Zur Kontrolle stehen ab und zu in Klammern die Anzahl von Wörtern, die ich selbst gefunden habe)!

PS: Im Anhang findet Ihr wieder viele wichtige Tips und Beispiele zum Umgang mit regulären Ausdrücken!

- Die Datei woerter.txt findet Ihr auf userem ftp Server im Verzeichnis blatt6
- Verwendet egrep immer mit der Option -E (für extended regular expression), damit alle Möglichkeiten von regulären Ausdrücken auch auf thales funktionieren!
- Sucht nach allen Wörtern, die das Textmuster „wiwi" enthalten
- Sucht nach allen Wörtern, die das Textmuster „bios" enthalten (2)
- Sucht nach allen Wörtern, die mit dem Textmuster „Mal" bzw. „mal" beginnen (32)
- Sucht nach allen Wörtern, die mit „mal" enden (62)

- Welche Wörter enthalten das Textmuster „oberon“
- Sucht nach allen Wörtern, die „ober“, gefolgt von einem beliebigen Zeichen, gefolgt von „n“ enthalten (6)
- Sucht nach allen Wörtern, die „ober“, gefolgt von beliebig vielen beliebigen Zeichen, gefolgt von „n“ enthalten (15)
- Wie kann man nach allen Wörtern suchen in denen das Textmuster „eb“ n-mal hintereinander vorkommt (die Lösung ist nicht n-mal hintereinander eb tippen). Sucht nach allen Wörtern, die eb 2 mal hintereinander enhalten, z.B. Hebebuehne
- Welche Wörter enthalten ganau 4 a’s, welche mindesten 4 a’s?
- Findet alle Wörter, die mit J beginnen und genau 2 a’s enthalten (4)

Aufgabe 2: Reguläre Ausdrücke für Spezialisten *(3 Punkte)*

- Die Datei names.txt findet Ihr ebenso wie woerter.txt im entsprechenden ftp Server Verzeichnis (siehe Aufgabe 1)
- Mit egrep bitte aus der Datei names.txt nur den Eintrag -Hans ausgeben lassen!
- Findet in der Datei woerter.txt alle Palindrome, die aus max. 5 Buchstaben bestehen. Palindrome sind Wörter, die vorwärts und rückwärts gelesen gleich sind, wie z.B. Radar.
- Findet in der Datei woerter.txt das Wort, in dem die 5 Vokale irgendwo in aufsteigender Reihenfolge vorkommen. Also ...a...e...i...o...u ist richtig, ...a...a...e...i...o...u...a z.B. auch, ...a...e...a...i...o...u nicht (Nicht zu viel Zeit in diese Aufgabe stecken)!
- Mit egrep bitte aus der Datei names.txt alle Einträge, die einen beliebigen Zwischenraum (Leerzeile, TAB oder beides) enthalten ausgeben (Wirklich nur für Spezialisten)!!!

Anhang: Kurzeinführung reguläre Ausdrücke

```
Ein Muster kann eine einfache Zeichenfolge sein, beispielsweise "abc".
Das Interessante an Mustern sind jedoch die Sonderzeichen und Ausdrücke,
mit denen man Muster flexibler gestalten kann. Im folgenden werden die
wichtigsten dieser Sonderzeichen und Ausdrücke vorgestellt.


Sonderzeichen zum Zählen

Mit diesen Ausdrücken kann angegeben werden, wie oft ein gewisses Zeichen
vorkommen darf oder muss. Der Ausdruck muss dabei immer direkt hinter dem
Zeichen stehen.

 ?: Null- oder einmal. Auf das Muster "ab?c" passen also "ac" und "abc",
    aber nicht "abbc".

 *: Null- oder mehrmals. Auf das Muster "ab*c" passen also "ac", "abc"
```

und "abbc", aber nicht "adc".

+: Ein- oder mehrmals. Auf das Muster "ab+c" passen also "abc" und "abbc", aber nicht "ac".

{n}: Genau n-mal. Auf das Muster "ab{1}c" passt also "abc", aber nicht "abbc". Das Muster ließe sich zu "abc" vereinfachen.

{n.m}: Zwischen n- und m-mal. Auf das Muster "ab{2,3}c" passen also "abbc" und "abbbc", aber nicht "abc".

{n,}: Mindestens n-mal. Das Muster "ab{1,}" ist also äquivalent zu "ab+c". Dieser Ausdruck kann auch vollständig durch vorherige Ausdrücke ersetzt werden; äquivalent zu "ab{5,}c" ist beispielsweise "ab{4}b+c".

Die Sonderzeichen zum Zählen müssen sich nicht nur auf ein einzelnes Zeichen beziehen. Wenn man mehrere Zeichen mit runden Klammern umgibt, werden diese als Gruppe behandelt (die Klammern sind hierbei Sonderzeichen, das Muster sucht also nicht nach den Zeichen "(" und ")"). Auf das Muster "(abc)+" passen also unter anderem "abc", "abcabc" und "abcabcabc".

Metazeichen

Innerhalb eines regulären Ausdrucks versteht man unter einem Metazeichen ein normales Zeichen, das durch einen vorangestellten Backslash (\) eine besondere Bedeutung erhält. JavaScript unterstützt die folgenden Metazeichen:

\b: Wortgrenze. An der Stelle, an der dieses Metazeichen steht, muss ein Wort beginnen oder aufhören, damit das Muster passt. Auf das Muster "\babc" passt beispielsweise "abcde", aber nicht "ababc". Auf "\babc\b" passt nur "abc"; das erste \b steht für den Wortanfang, das zweite für das Wortende.

\B: Keine Wortgrenze, das Gegenteil von \b. Auf das Muster "\Babc" passt beispielsweise "ababc", aber nicht "abcde". Auf das Muster "\Babc\B" passt weder "ababc" (Wortende) noch "abcde" (Wortanfang), aber "babcb".

\d: Ziffer. Auf das Muster "\d\d\d\d" passt beispielsweise "0815", aber nicht "R2D2".

\D: Keine Ziffer. Auf das Muster "\D\D" passt weder "12" noch "A1" noch "1A", aber "AB".

\s: Leerzeichen. Auf das Muster "Java\sScript" passt "Java Script", aber nicht "JavaScript".

\S: Kein Leerzeichen. Auf das Muster "Java\SScript" passen "Java_Script" und "Java-Script", aber nicht "Java Script".

\w: Buchstabe, Ziffer oder Unterstrich (_). Auf das Muster "\w" passen also beispielsweise "A", "a", "1" und "_", aber nicht "!".

\W: Kein Buchstabe oder Ziffer oder Underscore. Auf das Muster "\W" passt beispielsweise "!", aber nicht "A", "a" oder "_".

Weitere Ausdrücke

.: Jedes beliebige Zeichen außer einem Zeilensprung. Auf das Muster ".." passt also jede beliebige Folge zweier Zeichen.

[...]: Eine Auswahlliste von Zeichen. Aus der Liste von Zeichen kann genau eines ausgewählt werden. Die Zeichen stehen dabei hintereinander. Es können auch Abkürzungen vorgenommen werden, beispielsweise bezeichnet "[A-Z]" alle Großbuchstaben und "[0-9]" alle Ziffern. Auf das Muster "[LJ]ava" passt also beispielsweise "Lava" und "Java", aber nicht "Cava". Das Metazeichen "\w" kann durch folgendes Muster ausgedrückt werden: "[a-zA-Z0-9_]" (alle Klein- und Großbuchstaben, Ziffern und Unterstrich).

[^...]: Negierte Auswahlliste. Das Muster "[^a-zA-Z0-9_]" ist also äquivalent zum Metazeichen "\W".

^: Zeilenanfang. Auf das Muster "^Java" passen beispielsweise "Java" und "JavaScript", aber nicht "I love Java".

$: Zeilenende. Auf das Muster "Java$" passen beispielsweise "I love Java" und "Java", aber nicht "JavaScript".

\: Entwertung eines Sonderzeichens. Das Dollar-Symbol "$" hat in regulären Ausdrücken eine besondere Bedeutung. Will man aber nach dem Währungssymbol suchen, so muss man "$" mit einem vorangestellten Sonderzeichen entwerten. Also passen auf das Muster "\d\d\d \$" beispielsweise "100 $" und "200 $". Natürlich kann ein Backslash auch sich selbst entwerten; auf das Muster "\\" passt "\".

|: Oder-Operation. Auf das Muster "abc|def" passen unter anderem "abc" und "def".

Viel Erfolg!