1.8.V Anwendung/Suchen und Finden im Dateisystem – Versuch

1.8.V.1 Suchen mit Regulären Ausdrücken

In Deutschland besteht eine Postleitzahl aktuell aus fünf Ziffern, z.B. 12345. Mit folgendem Muster findet man alle Postleitzahlen in einer Datei: `'[0-9]{5}'` Jetzt stellt sich die Frage, was gefunden und was ausgegeben werden soll:

a) Fall 1: Jede Zeile, die eine Fundstelle enthält

Terminal
```
schueler@debian964:~$ egrep '[0-9]{5}' grepbsp.txt
01000
99998
33607 Bielefeld
05758016
#define PI 3.14159
05231
```

Man sieht, dass alle Kombinationen aus fünf aufeinanderfolgenden Ziffern gefunden werden. Eventuell sind auch andere Daten dabei.

b) Fall 2: Jede Fundstelle ohne den Rest der Zeile

Terminal
```
schueler@debian964:~$ egrep --only-matching '[0-9]{5}' grepbsp.txt
01000
99998
33607
05758
14159
05231
```

Auch hier sind andere Daten dabei. Aber es werden nur die Fundstellen angezeigt.

c) Fall 3: Jede Zeile, die nur eine PLZ enthält und sonst nichts

Terminal
```
schueler@debian964:~$ egrep '^[0-9]{5}$' grepbsp.txt
01000
99998
05231
```

Die Option `--only-matching` gibt hier keinen Sinn, weil keine sonstigen Daten in der Zeile vorhanden sind. Hier sind nun tatsächlich nur Zeilen dabei, die Postleitzahlen sein *könnten*. Ob es wirklich Postleitzahlen sind, hängt von der Art der Eingabedaten ab.

1.8.V.2 Beispiele für die Entwicklung Regulärer Ausdrücke

In den folgenden Beispielen soll immer der dritte Fall angenommen werden: Der zu suchende String steht alleine in einer Zeile. Deshalb muss jedes Muster mit `^` beginnen und mit `$` enden. Lässt man diese Ankerzeichen weg (oder ersetzt sie durch `\b`), so kann man die anderen Fälle abdecken.

a) Gesucht werden Zeilen mit einem IATA-Flughafen-Code.
Der IATA-Flughafen-Code besteht aus drei Großbuchstaben (z. B. RTM).
Lösung: `^[A-Z][A-Z][A-Z]$` oder `^[A-Z]{3}$`
Test der Lösung (mit der Datei `grepbsp.txt` aus `L18S.tgz`):

Terminal
```
schueler@debian964:~$ egrep '^[A-Z][A-Z][A-Z]$' grepbsp.txt
NBO
RTM
```

b) Gesucht werden Leerzeilen.
Lösung: `^$`
Test der Lösung:

Terminal
```
schueler@debian964:~$ egrep '^$' grepbsp.txt
  (Leerzeile)
```

c) Gesucht werden Zeilen, die genau ein Zeichen haben.
Hinweis: `.` steht für ein beliebiges Zeichen.
Lösung:

d) Gesucht werden Zeilen, die aus einem Punkt bestehen.
Hinweis: Der Punkt als gesuchtes Zeichen muss maskiert werden, damit er nicht *von grep* als Sonderzeichen interpretiert wird. Außerdem muss das ganze Suchmuster maskiert werden, damit seine Sonderzeichen nicht *von der Shell* interpretiert werden.
Lösung:

e) Gesucht werden alle nicht leeren Zeilen.
Hinweis: `.*` ist eine beliebige, auch leere Zeichenkette. Deshalb reicht `.*` nicht aus, um eine Zeichenkette zu beschreiben, die mindestens ein Zeichen hat.
Lösung:

f) Gesucht werden Zeilen mit dem „Amtlichen Gemeindeschlüssel“ in Deutschland.
Er kennzeichnet jede Stadt und jedes Dorf und besteht aus genau acht Ziffern.
Lösung:

g) Gesucht werden Zeilen mit der Kennung einer US-Rundfunkstation.
Rundfunkstationen in den USA haben eine Kennung aus vier Großbuchstaben. Das erste Zeichen ist ein W (Osten) oder ein K (Westen).
Lösung:

h) Gesucht werden Kommentarzeilen in C, C++ oder Java.
Eine Kommentarzeile beginnt mit `//`, danach können beliebige Zeichen folgen, die Kommentarzeile endet mit dem Zeilenende.
Lösung:

i) Gesucht werden Zeilen mit Telefonnummern.
Allgemeine Telefonnummern enthalten in beliebiger Reihenfolge die Zeichen
`0123456789 +-()/`, mindestens aber ein Zeichen.
Hinweis: Das Minuszeichen kann nur am Beginn oder Ende stehen, damit es nicht als Sonderzeichen (für einen Bereich) missverstanden wird (genauso könnte das Zeichen `]` nur am

Anfang stehen). Das Zeichen ^ dagegen darf nicht am Anfang stehen, wenn danach gesucht werden soll.
Lösung:

j) Gesucht werden die Pfadnamen von MP3-Dateien.
Verzeichnisanteile sollen erlaubt sein; der Pfadname soll die Endung `.mp3` haben.
Hinweis: Der Punkt muss maskiert werden, sonst würde z. B. auch `/media/mp3` als Name akzeptiert.
Lösung:

k) Gesucht werden Namen von HTML-Dateien.
Sie enden auf `.html` oder auf `.htm`.
Hinweis: Der Punkt muss maskiert sein.
Lösung:

l) Gesucht werden Kfz-Kennzeichen.
Kfz-Kennzeichen bestehen aus ein bis drei Großbuchstaben, einem Minuszeichen, ein bis zwei weiteren Großbuchstaben; danach folgen ein bis vier Ziffern (wovon die erste nicht null sein kann); optional folgen die Zusatzmerkmale `E` oder `H`.
Lösung:

m) Gesucht werden XML-Tags.
Sie stehen zwischen < und >.
Lösung:

n) Gesucht werden Strings, die kein Gänsefüßchen enthalten.
In C, C++ oder Java beginnt ein String mit Gänsefüßchen und endet auch damit. In diesem Fall soll im String selbst kein Gänsefüßchen vorkommen (abweichend von den Programmiersprachen).
Lösung:

o) Gesucht werden Strings.
In C, C++ oder Java beginnt ein String mit Gänsefüßchen und endet auch damit. Im String kann man ein Gänsefüßchen durch `\"` maskieren.
Lösung:

p) Gesucht werden Zeilen mit dem Beginn des Abschnitts einer ini-Datei.
Ein Abschnitt in einer ini-Datei beginnt mit einer öffnenden eckigen Klammer und endet mit einer schließenden eckigen Klammer. Dazwischen sind beliebige Zeichen erlaubt.
Hinweis: Die öffnende eckige Klammer muss maskiert werden.
Lösung:

q) Gesucht werden C++-Schlüsselworte.
Schlüsselworte in C++ sind `if`, `else` und `while` (es gibt noch mehr, aber für diesen Zweck sollen die drei reichen).
Hinweis: Der ODER-Ausdruck muss wieder geklammert werden.
Lösung:

r) Gesucht werden Zahlen aus $\mathbb{N}$:
Eine natürliche Zahl im Dezimalzahlensystem fängt mit eins bis neun an, danach sind beliebig viele Ziffern von null bis neun erlaubt.
Lösung:

s) Gesucht werden Zahlen aus $\mathbb{N}_0$:
Eine nichtnegative Zahl im Dezimalzahlensystem fängt mit eins bis neun an, danach sind beliebig viele Ziffern von null bis neun erlaubt. Außerdem gibt es die Zahl null.
Lösung:

t) Gesucht werden Zahlen aus $\mathbb{Z}$:
Eine ganze Zahl im Dezimalsystem hat optional zu Beginn ein Plus- oder Minuszeichen. Danach folgt der Betrag. Der Betrag ist eine nichtnegative Zahl.
Lösung:

u) Gesucht werden Zahlen aus $\mathbb{Q}$, dargestellt als Bruch:
Ein Bruch hat optional ein Vorzeichen, dann folgen der Zähler (Element aus $\mathbb{N}_0$), ein Schrägstrich (`/`) und der Nenner (Element aus $\mathbb{N}$).
Lösung:

v) Gesucht werden Zahlen aus $\mathbb{Q}$, entweder als Bruch oder als Ganzzahl dargestellt:
Ein Bruch hat optional ein Vorzeichen, dann folgen der Zähler (Element aus $\mathbb{N}_0$), ein Schrägstrich (`/`) und der Nenner (Element aus $\mathbb{N}$). Dabei soll der Schrägstrich mit Nenner optional sein, so dass auch eine ganze Zahl erkannt wird.
Lösung:

w) Gesucht werden Ortsnamen mit *mindestens* einem n darin.
Lösung:

x) Gesucht werden Ortsnamen mit *genau* einem n darin.
Lösung:

y) Gesucht werden Ortsnamen mit *höchstens* einem n darin.
Lösung:

z) Gesucht werden Hostnamen:
Der Hostname eines Systems soll mindestens zwei Punkte haben, die nicht direkt aufeinander folgen und nicht am Anfang oder Ende liegen, z. B. www.beispiel.de oder www.xyz.beispiel.de
Lösung: