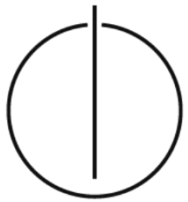

TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK
Praktikum: Grundlagen der Programmierung

Programmierpraktikum

Woche 02 (03.11.2017)

Stefan Berktold
s.berktold@tum.de

Präsenzaufgaben

Heutige Übersicht

Aufgabe 2.1 ***Kontextfreie Grammatiken***
Grammatik für ganze Zahlen (mit Unterstrichen, ohne führende Nullen)

Aufgabe 2.2 ***Reguläre Ausdrücke***
Erkennung bestimmter Textmuster

Aufgabe 2.3 ***Programmieraufgabe*** *(MiniJava)*
Würfelspiel: „Mäxchen“ / „Meiern“ (21)

Aufgabe 2.4 ***Programmieraufgabe*** *(MiniJava)*
Würfelspiel: „Lustige Sieben“

Hausaufgaben

Abgabefrist: 06.11.2017 05:00 Uhr

Aufgabe 2.5 [4] ***Kontextfreie Grammatiken***
Grammatik für Telefonnummern und E-Mail-Adressen

Aufgabe 2.6 [4] ***Reguläre Ausdrücke***
Erkennung bestimmter Textmuster

Aufgabe 2.7 [4] ***Programmieraufgabe*** *(MiniJava)*
Berechnung der Summe aller durch 3 oder 7 teilbaren Zahlen (bei geg. Maximum)

Aufgabe 2.8 [4] ***Programmieraufgabe*** *(MiniJava)*
Primzahl-Prüfer

Aufgabe 2.9 [4] ***Programmieraufgabe*** *(MiniJava)*
Generierung Latex-formatierter Tabellen von Potenzzahlen

Wiederholung

Kontextfreie Grammatiken (EBNF) / Reguläre Ausdrücke

Bekannte Operatoren:

- Alternative: `|` (entweder links oder rechts)
- Konkatenation: (<kein Zeichen> direkte Verknüpfung)
- Option: `?` (direkter Vorgänger keinmal oder einmal)
- Iteration: `*` (direkter Vorgänger keinmal, einmal oder beliebig oft)

Bindungsstärke

niedrig/schwach → hoch/stark

Alternative — Konkatenation — Iteration/Option

Ganze Zahlen (*mit* führenden Nullen):

```
Ziffer ::= 0 | … | 9
Zahl ::= (+|-)? Ziffer Ziffer*
```

Aufgabe 2.1

Grammatik Integer

Ab Java SE 7 können beliebig viele Unterstriche zwischen Ziffern in numerischen Literalen auftreten. Diese Unterstriche haben keine semantische Bedeutung und dienen lediglich der Lesbarkeit. Beispiel: `int x = 100_000;` wird interpretiert wie `int x = 100000;`.

In der Vorlesung wurde eine Grammatik für ganze Zahlen mit führenden Nullen angegeben. Geben Sie eine Grammatik für ganze Zahlen *ohne* führende Nullen und mit den ab Java SE 7 erlaubten Unterstrichen für numerische Literale an. Beachten Sie, dass die Unterstriche nur zwischen Ziffern, nicht aber am Anfang oder Ende eines numerischen Literals erlaubt sind, siehe `http://docs.oracle.com/javase/7/docs/technotes/guides/language/underscores-literals.html`.

Aufgabe 2.1 – Lösung

Grammatik Integer

```
NZiffer       ::=  1 | ... | 9                    // natürliche Ziffer
Ziffer        ::=  0 | NZiffer
Unterstrich   ::=  _*                             // beliebige Länge (auch 0)
Zahl          ::=  (+|-)? NZiffer (Unterstrich Ziffer)* | 0
```

Aufgabe 2.2

Reguläre Ausdrücke

Geben Sie unter Verwendung der in der Vorlesung eingeführten Operatoren für reguläre Ausdrücke *, ? und | einen regulären Ausdruck für a^+ an, wobei α^+ bedeutet, dass mindestens einmal und beliebig oft hintereinander ein durch den regulären Ausdruck α erzeugter Text auftritt.

Geben Sie reguläre Ausdrücke für die folgenden Textmuster an, wobei `letter` einen beliebigen Buchstaben im Text beschreibt:

1. Alle Texte, die kein b enthalten.
2. Alle Texte, die mit a oder b beginnen und mit c enden.
3. Alle Texte, die mit a beginnen und mit b enden oder mit b beginnen und mit a enden.

Es gilt `letter` $::= a|\ldots|z$.

Aufgabe 2.2 – Lösung

Reguläre Ausdrücke

Regulärer Ausdruck für `a+`: `a a*`

1. `(a | c | ... | z)*`
2. `(a|b) letter* c`
3. `(a letter* b) | (b letter* a)`

Wiederholung

Java-Konstrukte

- Variablendeklaration: <Typ> <Variablenname>;
- Zuweisung: <Variablenname> = <Ausdruck>;
- Bedingte Anweisung:

```
if (<Bedingung>) {
        <Statements>
} else if (<Bedingung>) {
        <Statements>
} else {
        <Statements>
}
```

- `while`-Schleifen:

```
while (<Bedingung>) {
        <Statements>
}
```

- Speziell für MiniJava: `read();` und `write(`<Text / Ausdruck>`);`

Eine genaue Beschreibung zur Grammatik/Syntax von MiniJava gibt es auf Moodle.

Aufgabe 2.3

Meiern (Auszug)

Schreiben Sie ein `MiniJava`-Programm, mit dem man „Meiern“ gegen den Computer spielen kann. Der Spieler soll abwechselnd über Dialogboxen über seine und die Würfe des Computers informiert werden.

Realisieren Sie diese Aufgabe Schritt für Schritt:

- Die Klasse `MiniJava` stellt Ihnen eine Methode dice() zur Verfügung, die Ihnen einen zufälligen Zahlenwert von 1 bis 6 liefert.
- Um eine Zahl zwischen 1 und 6 zu würfeln, verwenden Sie beispielsweise den folgenden Code: `int zahl; zahl = dice();`.
- Werfen Sie zwei Würfel und bestimmen Sie den Wert des Wurfes.
- Behandeln Sie die besondere Rangfolge der Würfe.

Die Spielregeln und weitere Schritte der Realisierung finden Sie in der Angabe auf Moodle!

Aufgabe 2.3 – Lösung

Meiern (Auszug)

Die offiziellen Lösungen zu allen Präsenzaufgaben sind auf *Moodle* verfügbar.

Meinen Lösungsvorschlag gibt es unter http://tutor17.stecrz.de.

Aufgabe 2.4

Lustige Sieben (Auszug)

In dieser Aufgabe wollen wir das Würfelspiel *Lustige Sieben* als ein Programm namens `LustigeSieben.java` schreiben. Es gibt einen Spieler, der gegen die Bank spielt. Der Spieler startet mit einem Guthaben von 100. Der Spieler setzt einen Teil seines Guthabens auf nur eines der Felder des Spielfelds. Die Bank würfelt mit zwei Würfeln. Dazu steht Ihnen die Methode `int dice()` der Klasse `MiniJava` zur Verfügung. Anschließend zahlt die Bank entsprechend folgender Regel an den Spieler dessen Gewinn aus:

- Der dreifache Einsatz wird ausgezahlt, falls die Summe der beiden Würfel 7 ergibt und der Spieler auf die 7 gesetzt hat;
- der doppelte Einsatz wird ausgezahlt, falls die Summe der beiden Würfel genau der gewählten Zahl des Spielfeldes entspricht;
- der einfache Einsatz wird zurückgezahlt, falls sich das Würfelergebnis auf derselben Längsseite wie die gewählte Zahl befindet.

7	
2	8
3	9
4	10
5	11
6	12

Weitere Informationen entnehmen Sie bitte der offiziellen Angabe auf Moodle!

Aufgabe 2.4 – Lösung

Lustige Sieben (Auszug)

Die offiziellen Lösungen zu allen Präsenzaufgaben sind auf *Moodle* verfügbar.

Meinen Lösungsvorschlag gibt es unter http://tutor17.stecrz.de.