Algorithmen und Datenstrukturen

5. Übungsblatt

Zeitraum: 17. – 21. November 2014

Ersatzveranstaltungen für die ausfallenden Übungen am Buß- und Bettag (19.11.):

- 1.DS: am Fr, 21.11., 1.DS, Raum APB/E01/U
- 4.DS: am Di, 18.11., 3.DS, Raum SCH/A/107
- 6.DS: am Do, 20.11., 1.DS, Raum APB/E01/U

Übung 1 (AGS 2.2.35)
Sei $\mathcal{E} = (V, \Sigma, S, R)$ mit $V = \{S, A\}$, $\Sigma = \{a, b\}$ und $R = \{\ S ::= [Aaa],\ A ::= (bbS \mid aa)\ \}$. Berechnen Sie die syntaktischen Kategorien $W(\mathcal{E}, S)$ und $W(\mathcal{E}, A)$ mit Hilfe der Fixpunktsemantik. Gehen Sie dazu in den folgenden Schritten vor:

(a) Dokumentieren Sie mindestens 5 Iterationsschritte,
(b) und schreiben Sie in Mengenschreibweise die Sprachen $W(\mathcal{E}, S)$ und $W(\mathcal{E}, A)$ auf.

Übung 2 (AGS 3.2.35 (c))
Bei einer Multiplikationstabelle für die Zahlen von 1 bis n enthält die Zelle in der i-ten Spalte und der j-ten Zeile den Wert $i \cdot j$, wobei $i, j \in \{1, \ldots, n\}$. So ist die Multiplikationstabelle für die Zahlen von 1 bis 5 z. B.:

·	**1**	**2**	**3**	**4**	**5**
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20
5	5	10	15	20	25

Schreiben Sie eine Funktion **void** table(**int** n), welche den Inhalt der Multiplikationstabelle für die Zahlen von 1 bis n ausgibt (mit printf). Der Inhalt der obigen Tabelle enthält die Zellen mit jenen Zahlen, die nicht fett gesetzt sind. Achten Sie auf die Ausgabe des Zeilenumbruchs zwischen den einzelnen Zeilen. In der obigen Tabelle sind die Zellen rechtsbündig ausgerichtet und stehen genau untereinander. Darauf können Sie in Ihrer Implementierung jedoch verzichten.

Übung 3 (AGS 3.1.8 ⋆)
Im folgenden ist die Ackermann-Funktion ack: $\mathbb{N}^2 \to \mathbb{N}$ gegeben. Implementieren Sie diese in C. Testen Sie Ihr Programm.

$$\begin{aligned} \mathrm{ack}(0, y) &= y + 1 && (y \geq 0) \\ \mathrm{ack}(x+1, 0) &= \mathrm{ack}(x, 1) && (x > 0) \\ \mathrm{ack}(x+1, y+1) &= \mathrm{ack}(x, \mathrm{ack}(x+1, y)) && (x, y > 0) \end{aligned}$$

Übung 4 (AGS 3.1.6 ⋆)
Schreiben Sie für die Berechnung von Fibonacci-Zahlen ein C-Programm, welches eine natürliche Zahl als Eingabe fordert und den zugeordneten Funktionswert ausgibt. Die Berechnung der

Fibonacci-Zahl selbst soll mit Hilfe einer Funktion `int fib(int z)` erfolgen. Realisieren Sie zwei Varianten der Funktion `fib` und machen Sie Aussagen zur Effizienz beider Programmvarianten:

(a) einen iterativ arbeitenden Algorithmus und

(b) einen rekursiv arbeitenden Algorithmus.

Zusatzaufgabe 1 (AGS 3.2.34 (c))
Die Summe aller natürlichen Zahlen kleiner oder gleich 16, die Vielfache von 3 oder von 5 sind, ist $3 + 5 + 6 + 9 + 10 + 12 + 15 = 60$. Implementieren Sie eine Funktion `int s(int n)`, welche für den Parameter n die Summe aller natürlichen Zahlen kleiner oder gleich n ausgibt, welche Vielfache von 3 oder von 5 sind! Tipp: n ist genau dann durch m teilbar, wenn n % m == 0 gilt.

Zusatzaufgabe 2 (AGS 3.2.30 (a))
Schreiben Sie ein C-Programm, welches Folgendes leistet. Der Benutzer wird zur Eingabe einer natürlichen Zahl aufgefordert. Dann gibt das Programm der Größe nach alle ganzen Zahlen bei Null beginnend aus, die kleiner sind als die eingegebene Zahl.

Dabei gibt es folgende Ausnahmen: Wenn eine Zahl durch 3 teilbar ist, wird nur „ping“ ausgegeben. Wenn eine Zahl durch 4 teilbar ist, wird nur „pong“ ausgegeben. Wenn eine Zahl durch 3 und 4 teilbar ist, wird nur „pingpong“ ausgegeben.

Zusatzaufgabe 3 (AGS 3.1.14 (a))
Gegeben seien die Vorschriften zur Berechnung der folgenden durch c_x und c_y parametrisierten, voneinander abhängigen, reellen Zahlenfolgen x_i und y_i, $i \geq 0$:

$$x_0 = 0\,, \qquad y_0 = 0\,,$$
$$x_{i+1} = c_x + x_i^2 - y_i^2\,, \qquad y_{i+1} = c_y + 2 \cdot x_i \cdot y_i\,.$$

Schreiben Sie ein iteratives Programm, welches die Werte für c_x und c_y einliest und das kleinste $i \leq 1000$ ausgibt, sodass $x_i^2 + y_i^2 \geq 4$ gilt. Gilt diese Bedingung für kein $i \leq 1000$, dann soll „`>1000`“ ausgegeben werden.