Philipps-Universität Marburg
Fachbereich Mathematik und Informatik
Arbeitsgruppe Datenbanksysteme

Wintersemester 2013/2014
Montag, 4. November 2013

Prof. Dr. Bernhard Seeger
Dipl.-Inf. Marc Seidemann
Johannes Drönner, M.Sc.

Übungen zur Vorlesung

Praktische Informatik I

Blatt 3

Abgabe: Montag, 11.11.2013,
bis **spätestens** 10:00 Uhr
über die ILIAS-Plattform

Aufgabe 3.1: max-4 (4+3P=7P)

a) Implementieren Sie die Funktion `static double max4(int n, int m, int o, int p)` (analog zu `max3`, siehe Skript Folie 134).

b) Geben Sie die Speicherbelegung bei der Ausführung von `max4(1,2,3,4)` bis zum Ende der Methode an. Orientieren Sie sich dabei am Skript (Folie 135ff).

Aufgabe 3.2: Matrjoschka-Puppen iterativ (6P)

In der Vorlesung wurde ein rekursiver Algorithmus vorgestellt, mit dessen Hilfe eine massive Matrjoschka-Puppe aus ihren umgebenden hohlen Puppen herausgeholt werden kann (siehe Skript Folie 141).

In dieser Aufgabe sollen Sie den rekursiven Algorithmus mit Hilfe einer WHILE-Schleife in eine iterative Form bringen. Beschreiben Sie Ihren Algorithmus textuell.

Aufgabe 3.3: Rekursive Ausdrücke (6P)

Auf dem ersten Präsenzblatt im Tutorium haben wir eine Grammatik für einfache mathematische Ausdrücke betrachtet. Diese enthielt rekursive Beziehungen, z.B. für die Klammerung eines Ausdrucks:

<Ausdruck> ::= (….
| “(“ <Ausdruck> ”)”
| ….
)

Entwickeln Sie nun ein rekursives Schema, mit dem man geklammerte Ausdrücke in einem Programm auswerten kann. Konstruieren Sie analog zum Matrjoschka-Beispiel im Skript eine Methode `evaluateExpression(Expression e)` in *Pseudocode*[1].

Aufgabe 3.4: Pi rekursiv (7+3=10P)

a) Implementieren Sie die Annäherung von Pi aus Aufgabe 2.6 rekursiv. Lagern Sie dazu die Berechnung der Quadratbruchsumme (WHILE-Schleife) in eine rekursive Methode `static double quadraticFractionSum(double counter)` aus. Die Näherung soll wieder nach der 1000-ten Iteration abbrechen.

Hinweis:
Um Rundungsfehler zu vermeiden, ist die Variable `counter` vom Typ `double`.

b) Erläutern Sie, warum die Variante in Aufgabe 2.6 der Implementierung in Aufgabenteil a) vorzuziehen ist.

[1] Pseudocode dient dazu, eine Methode informell zu Beschreiben. Ein Algorithmus in Pseudocode muss also kein ausführbares Programm sein, sondern kann auch textuelle Beschreibungen enthalten. Siehe Matrjoschka-Beispiel aus der Vorlesung (Skript Folie 141).

Aufgabe 3.5: Binäre Suche (6+2=8P)

In dieser Aufgabe betrachten wir Rekursion im Alltag. Wir wollen die englische Übersetzung für das Wort „Haus“ in einem großen Wörterbuch finden. Ein einfacher Ansatz, das Wort zu suchen, wäre, das Buch von vorn nach hinten durchzublättern, bis wir das Wort Haus gefunden haben. Das macht Sinn, wenn wir annehmen, dass H relativ weit vorn im Wörterbuch auftaucht (weil es auch recht früh im Alphabet auftritt). Leider wissen wir aber nichts über die Verteilung der Anfangsbuchstaben der Wörter. Es könnte genauso gut sein, dass H erst sehr spät im Wörterbuch auftaucht, da nur wenige Wörter mit den Anfangsbuchstaben H, I, J,… existieren.

a) Nehmen Sie an, dass Sie keine Kenntnis über die Verteilung der Anfangsbuchstaben haben. Beschreiben Sie einen rekursiven Algorithmus, mit dessen Hilfe Sie möglichst schnell diejenige Seite im Wörterbuch finden, auf der das gesuchte Wort steht. Beginnen Sie damit, das Buch in der Mitte aufzuschlagen.

b) Wie könnte man den Algorithmus aus Aufgabenteil a) verbessern, wenn man die genaue Verteilung der Anfangsbuchstaben aller Wörter kennt.