FAU, Informatik 2, AUD-Team
aud@i2.cs.fau.de

1. Übung

Abgabe bis 08.05.2017, 10:00 Uhr

Bitte beachten:

Generelle Hinweise zu den Aufgaben und deren Bearbeitung finden Sie auf dem „organisatorischen Übungsblatt“. Sie können die Aufgaben erst im EST abgeben, nachdem Sie in die Tafelübungen eingeteilt wurden (voraussichtlich Sonntag).

Einzelaufgabe 1.1: Zahlensysteme und Codierung — 19 EP

Geben Sie die Ergebnisse dieser Aufgabe als `Darstellungen.pdf` über EST ab.

a) Ergänzen Sie die folgende Tabelle, indem Sie die gegebenen Werte in das entsprechende Zahlensystem umwandeln. Verwenden Sie bei negativen Zahlen **immer** die *Zweierkomplement-Darstellung*. Die Breite der Zahlen sei dabei immer 16 bit.

Dezimal	Binär	Oktal	Sedezimal
12852	?	?	?
-4711	?	?	?
?	00011100 01110001	?	?
?	11100011 10001110	?	?
?	?	010101	?
?	?	123456	?
?	?	?	6789
?	?	?	FEED

b) Codieren Sie den Text **„`AuD1$tT%\|`“** (ohne Anführungszeichen) als ASCII-Zeichen im Sedezimalsystem.

c) Geben Sie die Zeichenkette an, die sich hinter folgenden ASCII-Codes verbirgt:
$48_{(10)}$ $4D_{(16)}$ $107_{(8)}$ $00100001_{(2)}$

Gruppenaufgabe 1.2: Feld-Funktionen — 19 GP

Felder sind äußerst wichtige Datenstrukturen, dennoch sind sie in Java nicht ganz intuitiv zu benutzen. Erstellen Sie eine Klasse `ArrayFun` mit den folgenden drei statischen Klassenmethoden:

a) Die Methode `allocate` soll ein 3-dimensionales quaderförmiges **String**-Feld mit den „Kantenlängen“ $xDim \times yDim \times zDim$ erstellen und jeden Eintrag mit seinen Koordinaten im Feld wie im folgenden Beispiel vorbelegen:
`r = allocate(47, 11, 42)` $\Longrightarrow$ `r[46][10][41] == "(46,10,41)"`

b) Die Methode `stuff` erhält ein 4-dimensionales Feld `f`, soll dieses wie in der vorangehenden Teilaufgabe befüllen und anschließend zurückgeben. Vorhandene Einträge (egal ob `null` oder echte String-Objekte) sollen dabei überschrieben werden. Beachten Sie hierbei, dass `f` nicht symmetrisch sein muss, d.h. einzelne Unterfelder könnten unterschiedliche Länge haben oder sogar `null` sein. Hinweis: Sie dürfen in dieser Aufgabe *keine* neuen (Unter-)Felder anlegen; befüllen (d.h. überschreiben) Sie *nur* die vorhandenen Stellen mit entsprechenden **Strings**!

c) Die Methode `convolve` schließlich erhält ein 3-dimensionales Feld `f` und soll die Daten in ein 2-dimensionales Feld `r` „zusammenpressen“, d.h. sie soll alle vorhandenen Einträge ($\neq$ `null`) der dritten Dimension in `f` verketten und in der zweiten Dimension in `r` ablegen. Hinweis: Gibt es in einer Dimension von `f` keinen einzigen von `null` verschiedenen Eintrag, dann bleibt die zugehörige Dimension in `r` ebenfalls `null`.

Geben Sie Ihre Lösung als `ArrayFun.java` über EST ab.

Gruppenaufgabe 1.3: Babylon — 22 GP

Um $\sqrt[k]{a}$ näherungsweise zu berechnen, kann man u.a. das *babylonische Wurzelziehen* anwenden. Dazu bestimmt man die Nullstelle der Funktion $f(x) = x^k - a$ mit folgender Iterationsvorschrift:

$$x_{n+1} = \frac{(k-1) \cdot x_n^k + a}{k \cdot x_n^{k-1}}$$

Beginnend mit einem geeigneten Startwert x_0 wird die obige Gleichung solange iteriert, bis das Ergebnis hinreichend genau ist, d.h. bis zwei aufeinanderfolgende Werte um nicht mehr als eine vorgegebene Schranke ϵ voneinander abweichen: $|x_{n+1} - x_n| \leq \epsilon$.
In dieser Aufgabe gelten „vereinfachend“ folgende Vorgaben:

- Ist der Wurzelexponent k nicht positiv, dann soll das Ergebnis grundsätzlich -1 sein.
- Ist der Radikand a negativ, aber der Wurzelexponent k gerade, dann soll das Ergebnis ebenfalls -1 sein (keine komplexen Zahlen).
- Ist der Radikand a negativ, dann soll der Startwert $x_0 = -2$ gewählt werden; andernfalls soll mit $x_0 = 2$ begonnen und die nichtnegative Wurzel ermittelt werden.
- Sie dürfen weder die Betragsfunktion ($|x|$), noch die Wurzel- ($\sqrt[n]{x}$) oder Potenzfunktionen (x^n) verwenden – das gilt sowohl für die Theorie- als auch für Praxisteilaufgabe!

Im Folgenden soll eine entsprechenden Funktion/Methode `wurzel(a,k,ϵ)` entwickelt werden:

a) Beschreiben Sie den Algorithmus der Methode `wurzel` in *Pseudocode*. Verwenden Sie dazu ausschließlich die deutschen Pseudo-Anweisungen, die Sie in der Vorlesung (F. 1-24) kennengelernt haben (*kein* Java-Code!).

b) Stellen Sie nun den Algorithmus als *Ablaufdiagramm* (*nicht* Syntaxdiagramm!) dar. Verwenden Sie dazu die Darstellung, die Sie in der Vorlesung (F. 3-14, 3-37) kennengelernt haben.

c) Erstellen Sie nun eine Klasse `Babylon` mit genau einer statischen *Klassenmethode* der Form `double wurzel(double a, int k, double eps)`, die das obige Verfahren umsetzt.

Geben Sie Ihre Lösung als `Babylon.pdf` bzw. `Babylon.java` über EST ab.

19 EP + 41 GP = 60 Punkte