Übungsbeispiele Programmierpraktikum

Ferenc Domes

30. April 2008

1. (Algorithmen, Struktogramme) Erstellen Sie mit Hilfe von EasyCode ein Struktogramm, welches einen Einkauf in einem Supermarkt beschreibt.
   Anmerkungen für Bsp. 1-3:
   - Fangen Sie mit einem einfachen Algorithmus an (wenige Schritte).
   - Stellen Sie sicher, dass die Bedingungen Endlichkeit, Eindeutigkeit, Interpretierbarkeit für den Algorithmus erfüllt sind.
   - Verfeinern Sie den Algorithmus durch Hinzufügen zusätzlicher Schritte.
   - Testen Sie wieder die obigen Bedingungen.
2. (Algorithmen, Struktogramme II) Erstellen Sie mit Hilfe von EasyCode ein Struktogramm, zur Bedienung eines Kaffeeautomaten.
3. (Algorithmen, Struktogramme III) Erstellen Sie zwei Algorithmen zum Finden einer geeigneten Freundin bzw. Ehefrau oder eines geeigneten Freundes bzw. Ehemanns. Die Schritte in Aufgabe 1 sollten für beide Algorithmen erneut durchgeführt werden. Algorithums 1 sollte Ihrer Meinung nach einen sehr guten Weg zum Erlangen des Ziels darstellen, und Algorithums 2 einen völlig ungeeigneten.
4. (Java-Grundlagen, Operatoren) Berechnen Sie am Papier (Schreibtischtest) die Werte der Variablen `i`, `j`, `k`, `zahl`, `test1` und `bitw` anhand des Codes:

```
int i=1;
int j=(int)8.9;
boolean test1= (--i<=1) || (j != 8.9);
boolean test2= !test1;
int k=j<<i;
i=k++;
j=--k*i;
double zahl=(test1 || (j>-5)) ? 5.99:-88;
test2=test1 && (zahl != -88);
int bitw=(--i<=6) ? 32:15;
bitw%=2;
bitw=~bitw;
```

   Implementieren Sie ein Programm (Bsp04), das die obige Operationen ausführt und die gesuchten Werte ausgibt. Vergleichen Sie Ihre Ergebnisse und die Resultate, die vom Programm erzeugt wurden. Falls diese nicht übereinstimmen, suchen Sie so lange nach Fehlern in ihrem Schreibtischtest, bis die Ergebnisse identisch sind.

5. (Eingabe, Operatoren II) Erweitern Sie das vorherige Beispiel so, dass die Zahl `i` und die Zahl `j` mittels `Read.java` eingelesen und alle Resultate anhand der eingelesenen Werte berechnet werden.

6. (Bedingungen, Schleifen) Folgender Ablauf soll mit einem Struktogramm, einem Schreibtischtest (inkl. Darstellung und Umrechnung des internen Zahlenformates) und einem Java-Programm dargestellt werden. Die Ergebnisse der Zahlenoperationen sollen manuell überprüft werden. Es soll eine Schleife 5x durchlaufen werden. Der Wert, beginnend mit 10 wird nach jedem Durchlauf um 1 erhöht. Ist die daraus resultierende Zahl gerade, soll eine Meldung ausgegeben werden. Ist die Zahl durch 3 teilbar, so werden eine Meldung ausgegeben und zusätzlich folgende Operationen mit dieser Zahl durchgeführt: Bitweise Invertierung, Linksshift um 3 Stellen. Die Ergebnisse sollen angezeigt werden.

7. (Eingabe, Bedingungen) Erstellen Sie eine Applikation, die max. 15 Integer-Zahlen (Werte zwischen 0 und 20) von der Tastatur einliest. Fehlerhafte Eingaben werden mit einer Fehlermeldung angezeigt. Die eingelesenen Zahlen sollen in einer einfachen Grafik folgendermaßen dargestellt werden: Pro Zahl wird in einer Spalte (!) der Wert der Zahl als Sternchen-Balken ($*$) und der Wert selbst dargestellt. Der Benutzer muss allerdings nicht alle 15 Zahlen angeben; die Eingabe kann mit `x` vorzeitig beendet werden. Die restlichen, nicht eingegebenen Zahlen werden als 0 in der Grafik angezeigt. Mit Struktogramm und Schreibtischtest. Schreibtischtest lediglich für 2 gültige und 2 ungültige Zahlen durchführen, Hilfestellung für Lesen, Umwandlung und Testen der Eingabe: `Read.java`

8. (Eingabe, Sortierung) Erstellen Sie eine Klasse, welche eine beliebige Anzahl von Integer-Zahlen nach dem Bubble-Sort-Verfahren (Sortierung durch fortlaufende Vertauschung von benachbarten Zahlen) sortiert.

   - Der Algorithmus muss selbst implementiert (=erstellt) werden. Die Verwendung z. Bsp. der sort()-Methode der Klasse java.util.arrays und dergl. ist nicht gestattet.
   - Die Zahlen werden als ein String eingelesen. Die Zahlen sind durch Komma getrennt. Beispiel $5, 70, 4, 6, 45$. Hinweis: Mit Hilfe der split()-Methode der Klasse java.lang.String kann der eingelesene String in eine Tabelle zerlegt werden.
   - Die Eingabe und der Sortiervorgang sollen in separaten Methoden gekapselt werden. Die Methoden sollen geeignete Parameter und Rükgabewerte enthalten, da es für dieses Beispiel nicht gestattet ist, Daten mittels globaler Variablen zu übergeben.
   - Nach dem Sortiervorgang muss die sortierte als auch die unsortierte Tabelle ausgegeben werden.
   - Erstelle ein Struktogramm für den Sortiervorgang (die Eingabe kann, muss aber nicht im Struktogramm abgebildet werden) Erstelle einen Schreibtischtest für mindestens 4 unsortierte Zahlen.

9. (Sortierungsverfahren)* Finden und implementieren Sie zwei andere Sortierungsverfahren. Erklären Sie am Beginn des Programmcodes was die Vorteile und Nachteile dieser Verfahren sind und wann sie verwendet werden sollten.

10. (Objekte, Klassen) Erstellen und erweiteren Sie ein Klassenmodell, welches die Objekte der `Personalverwaltung` abbildet. Z. Bsp. Person, Angestellter, Adresse,... (siehe Skriptums (Kap. 5.10)). scannen Sie die lesbar(!) geschriebene Lösung ein.

11. (Objekte, Mathematik) Schreiben Sie eine Klasse names `Matrix`, die die wichtigsten Matrix-Operationen implementiert. Die Klasse soll allgemeine $m \times n$ Matrizen mint `double` Einträgen ermöglichen. Folgende Methoden sollten im `Matrix` enthalten sein:

```
//Default constructor
public Matrix()
-------------------------------------------------------------------------
//Constructor generating a matrix of dimension d
public Matrix(Dimension d)
-------------------------------------------------------------------------
//Get the dimension of the matrix
public Dimension dim()
-------------------------------------------------------------------------
//Get the (c,r)th element of the matrix
public double get(int c, int r)
-------------------------------------------------------------------------
//Set the (c,r)th element of the matrix as d
public void   set(int c, int r, double d)
-------------------------------------------------------------------------
//Matrix addition
public Matrix add (Matrix a)
-------------------------------------------------------------------------
//Matrix-Scalar multiplication
public Matrix mult (double a)
-------------------------------------------------------------------------
//Matrix-Matrix multiplication
public Matrix mult (Matrix a)
-------------------------------------------------------------------------
//Sub-matrix generated by deleting the c-th column and the r-th row
public Matrix submatrix(int c, int r)
```

(*) Wenn Sie genug Lust und Zeit aufbringen können, schreiben Sie die `Matrix`-Klasse im sparse-Format, das heisst jedes von 0 verschiedene Element wird als ein Tripel (i,j,d) gespeichert, wobei der Integer i den Zeilenindex, der Integer j den Spaltenindex und der Double d den entsprechenden Eintrag symbolisiert. Implementieren Sie die obigen Methoden für sparse-Matrizen. Dieses Format ist sinnvoll, um Matrizen mit sehr wenigen von 0 verschiedene Eintrge speichersparend abzuspeichern.

12. (Rekursion, Mathematik) Implementieren Sie eine einfache Rekursion, die die Fakultät einer Zahl ausrechnet, und zeichnen Sie genau auf, was bei der Berechnung von 5! passiert. Schreiben Sie Ihr Protokoll in Word oder scannen Sie die lesbar(!) geschriebene Lösung ein.

13. (Rekursion, Mathematik)* Ergänzen Sie die Klasse `Matrix`, mit einer zusätzlichen rekursiven Funktion `private double det(Matrix d)`, die durch Verwendung den Laplaceschen Entwicklungssatzes die Determinante der Matrix `d` berechnet. Die Funktion `public double det()` sollte die obige rekursive Funktion verwenden um die Determinante der aktuellen Matrix zu bestimmen.
Der Laplaceschen Entwicklungssatz lautet:

Für $A \in \mathbb{R}^{n \times n}$ und $i \in 1 \ldots n$ beliebig fix gewählt, gilt dass

$$\det A = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij}).$$

A_{ij} bezeichnet eine $(n-1) \times (n-1)$ Untermatrix von A, die durch das Streichen der $i$ten Zeile und $j$ten Splate entsteht. Der Ausdruck a_{ij} symbolisiert den (i,j)ten Eintrag von A.
Den Index i können Sie beliebig wählen, und die Untermatrix A_{ij} bekommen Sie durch die schon im vorherigen Beispiel programmierte Methode `submatrix`.

14. (Numerische Lineare Algebra)**

    (a) Bestimmen Sie die Eigenwerte der Matrix. Implementieren Sie ein Symmetrie-Test.

    (b) Für positive definite, symmetrische, $n \times n$ Matrizen implementieren Sie die Cholesky-Zerlegung.

    (c) Besimmen Sie die QR-Zerlegung einer Matrix. Dazu können Sie die

    - Householdertransformationen,
    - Givens-Rotationen,
    - Gram-Schmidtsches Orthogonalisierungsverfahren,

    verwenden.

15. (Objekte, Klassen II) Implementieren Sie das Beispiel `Personalverwaltung` des Skriptums (Kap. 5.10) mit Hilfe eines Interfaces. Die Klasse Angestellter, Abteilungsleiter und Projektmitarbeiter sollen nicht voneinander abgeleitet werden. Hinweis: siehe Klassendiagramm `Angestellter.pdf`, das Laden der Tabelle kann auch in der Klasse erfolgen, die Daten brauchen nicht von der Konsole eingelesen werden. Das Laden und Drucken der Angestellten-Tabelle soll in separaten Methoden gekapselt werden. Die Datenübergabe zwischen den Methoden muss mittels geeigneter Parameter und Rückgabewerte implementiert werden. Globale Variablen sind für diesen Zweck nicht gestattet. Struktogramm und Schreibtischtest nicht erforderlich.

16. (Applets, Sortierung) Erstellen Sie ein Java-Applet, welches Zahlen sortiert. Die Zahlen werden mit einem JTextField als String und durch Komma getrennt eingelesen. Der Sortiervorgang kann vom Benutzer auf 2 Arten gestartet werden:

    - Klick auf einen Button
    - Drücken der ENTER-Taste im Eingabefeld.

    Die unsortierten und sortierten Zahlen sollen jeweils in einem JLabel angezeigt werden. Dazu soll eine Sortier-Methode aus Beispiel 8 oder 9 verwendet werden d.h. der Source-Code (.java Datei) soll in das neue Verzeichnis kopiert und adaptiert werden. Die Methoden können an die Erfordernisse des Applets angepasst werden. Die ursprüngliche Möglichkeit, Zahlen von der Konsole einzulesen, muss dabei erhalten bleiben.

17. (Applets, Layout, Buttons) Ein `JApplet` mit einem `BorderLayout` sollte die folgende Komponente enthalten:
    • Ein `JPanel` im Norden mit einen `GridbagLayout`. Darauf sollten folgende Komponenten geklebt und schön geordnet werden:

-ein `JLabel` 'Zeilenzahl:',
-ein `JTextField` für die Eingabe der Zeilenzahl (`m`),
-ein `JLabel` 'Spaltenzahl:',
-ein weiteres `JTextField` für die Eingabe der Spaltenzahl (`n`),
-ein `JButton` 'erzeugen'.
• Wenn der `JButton` 'erzeugen' gedrückt wird, sollte im Zentrum ein `JPanel` mit einem `m` mal `n` `GridLayout` erzeugt werden. Auf diesem `JPanel` sollten `m*n` durchnumerierte `JButtons` plaziert sein.
-Drückt man einen `JButton` mit gerader Nummer, werden dieser und alle benachbarten Buttons deaktiviert.
-Drückt man einen `JButton` mit ungerader Nummer wird die Farbe seiner Zahl auf rot geändert und alle benachbarten Buttons aktiviert.
-Drückt man einen `JButton` mit ungerader Nummer und roter Farbe werden alle deaktivierte Buttons aktiviert und alle aktivierte Buttons deaktiviert. Weiters setzt man alle Farben wieder auf schwarz.
• Im Süden soll ein `JButton` sein mit einem beliebigen Bild darauf. Dazu erzeugen Sie einen `ImageIcon` mit dem Konstruktor `(String filename)` und dann benutzen Sie dieses `ImageIcon` im Konstruktor `(String text, Icon icon)` vom `JButton`, um den gewünschten Button zu erhalten.

18. (Applets, Zeichnen)* Erstellen Sie ein Java-Applet, welches ein Rechteck mittels Klicken, Ziehen und Loslassen der Maus darstellt. Beim Klicken der Maustaste soll die linke obere Ecke des Rechtecks dargestellt werden. Sobald der User die Maus bei gedrückter Maustaste zieht, soll an der aktuellen Cursorposition die rechte untere Ecke darggestellt werden. D.h. das Rechteck verändert seine Größe, solange der User die Maus bewegt. Zusätzlich sollen die Koordinaten des rechten oberen und linken unteren Eckpunkts sowie die Länge und Breite des Rechtecks dargestellt werden.