HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Heinrich-Heine-Universität Düsseldorf
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik, Abteilung Betriebssysteme
Prof. Dr. Michael Schöttner
Kevin Beineke
Ludmila Himmelspach
Stefan Nothaas

4. Übungsblatt „Programmierung“ im Wintersemester 2016/17

Abgabe bis 14. November 2016, 07:59 Uhr über das Abgabesystem

Ab diesem Übungsblatt gibt es Punktabzug bei fehlender Überprüfung von Parametern, die über das Terminal eingelesen werden (z.B. abweichende Parameteranzahl übergeben). Ungültige Werte sollen außerdem allgemein frühstmöglich abgefangen werden.

Aufgabe 1: Operationen mit 2D Vektoren — 10 Punkte

In dieser Aufgabe sollen Sie Methoden für typische Vektor-Operationen implementieren. Dabei soll ein zwei-dimensionaler Vektor $\vec{v}$ mit den Komponenten x und y durch ein Array aus zwei primitiven double Werten dargestellt werden. Implementieren Sie die in der Vorgabe `Vec2D.java` mit *TODO* markierten Methoden. Rufen Sie in der `main` Methode die von Ihnen implementierten Funktionen auf und testen Sie diese.

- `create`: Erzeugt ein Array für zwei Elemente und speichert die Werte x und y in diesem.
- `add`: Addiere zwei Vektoren. Die Vektoraddition für die Vektoren $\vec{v}_1$ und $\vec{v}_2$ ist folgendermaßen definiert: $\begin{pmatrix} x_{\vec{v_1}} + x_{\vec{v_2}} \\ y_{\vec{v_1}} + y_{\vec{v_2}} \end{pmatrix}$
- `dot`: Berechnet das Skalarprodukt zweier Vektoren: $\vec{v}_1 \cdot \vec{v}_2 = x_{\vec{v_1}} \cdot x_{\vec{v_2}} + y_{\vec{v_1}} \cdot y_{\vec{v_2}}$
- `enorm`: Berechnet die Euklidische Norm $|\vec{v}|$ eines Vektors $\vec{v}$. Dies kann mithilfe des Skalarprodukts bestimmt werden: $|\vec{v}| = \sqrt{\vec{v} \cdot \vec{v}}$
- `normalize`: Normiert den gegebenen Vektor (d.h. die Euklidische Norm wird 1). Die Methode `normalize` erzeugt einen neuen Vektor dessen Komponenten auf die passend umgerechneten Werte des alten Vectors gesetzt werden. Für einen übergebenen Vektor $\vec{v}$ und dessen E-Norm n ergibt sich also der neue Vektor aus: $\begin{pmatrix} x_{\vec{v}}/n \\ y_{\vec{v}}/n \end{pmatrix}$

Aufgabe 2: Sortieren und Suchen — 20 Punkte

Schreiben Sie ein Program `SortAndSearch`, welches den Sortieralgorithmus BubbleSort sowie die Binäre Suche implementiert. Die Parameter für die Größe des Arrays und den zu suchenden Wert sollen als Argumente vom Terminal/Konsole übergeben werden. Gehen Sie hierfür folgendermaßen vor:

- Implementieren Sie eine Methode `sort`, welche ein beliebiges int-Array mit dem Sortierverfahren BubbleSort **absteigend** sortiert. Testen Sie Ihre Implementierung mit verschiedenen Arrays, die Sie mit Zufallszahlen füllen, und überprüfen Sie nach der Sortierung, ob das Array auch vollständig absteigend sortiert wurde. Das Ergebnis des Tests (Array sortiert oder nicht sortiert) soll auf dem Terminal ausgegeben werden.
- Implementieren Sie eine Methode `search`, welche ein bereits absteigend sortiertes Array und den zu suchenden Integer-Wert als Parameter übergeben bekommt. Die Methode soll -1 zurückgeben, falls der Wert im Array nicht gefunden wurde, andernfalls die Position des Wertes im Array. Falls der Wert gefunden wurde, so soll dessen Position auf dem Terminal ausgegeben werden. Falls dieser nicht gefunden wurde, soll eine für den Nutzer verständliche Nachricht ihm dies mitteilen.

Aufgabe 3: 2D Arrays — 20 Punkte

Implementieren Sie die in der Vorgabe `Array2D.java` mit `TODO` versehenen Methoden wie folgt:

- `createArray1D`: Erzeugt ein 1D-Array mit der übergebenen Dimension `size` und initialisiert dessen Inhalt vollständig vom nullten Index angefangen mit den aufsteigenden Werten 0, 1, 2, 3, ...
- `createArray2D`: Erzeugt ein 2D-Array mit den übergebenen Dimensionen `sizeX` und `sizeY`. Dessen Inhalt wird vollständig und zeilenweise vom nullten Index mit den aufsteigenden Werten 0, 1, 2, 3, ... initialisiert.
- `convert2DArrayTo1D`: Erzeugt ein 1D-Array aus einem 2D-Array. Das 1D-Array soll zeilenweise aus dem 2D-Array erstellt werden. Falls das 2D-Array nicht in das 1D-Array passt, so wird an dessen Grenze passend abgeschnitten. Bei zu wenigen Elementen wird mit dem Wert -1 aufgefüllt.
- `convert1DArrayTo2D`: Erzeugt ein 2D-Array aus einem 1D-Array, wobei der zusätzliche Parameter `sizeY` die Anzahl der Spalten angibt. Falls das 1D-Array nicht passend auf ein 2D-Array übertragen werden kann, so ist aufzurunden und die überschüssigen Zellen sind mit dem Wert -1 zu versehen. Falls das 2D-Array zu klein ist, so wird an dessen Grenze passend abgeschnitten.
- `print1D`: Gibt ein 1D-Array vollständig auf der Kommandozeile aus.
- `print2D`: Gibt ein 2D-Array vollständig und zeilenweise auf der Kommandozeile aus.

In der `main`-Methode ist bereits ein Test mit den zu erwartenen Ausgaben für ein Beispiel vorgegeben. Nutzen Sie diesen Test um damit sicherzustellen, dass Ihre implementierten Methoden auch korrekt funktionieren und die gegebenen Ausgaben liefern. Dieser Test deckt nicht alle möglichen Fälle ab. Erweitern Sie diesen gegebenfalls um weitere Fälle zu überprüfen.
Hinweise: Gehen Sie davon aus, dass Ihre Arrays immer mind. ein Element beinhalten.

Aufgabe 4: Transponierte Matrix — 10 Punkte

Schreiben Sie ein Programm `Matrix`, dass eine 3x3 Matrix transponiert.
Die Werte für die Matrix (natürliche Zahlen) sollen von der Standardeingabe eingelesen werden. Nutzen Sie für das Einlesen der Parameter passende Methoden aus der Klasse `StdIn`. Die beiden von uns bereitgestellten Dateien zu den Klassen `StdIn` und `StdOut` müssen sich beim Kompilieren im gleichen Verzeichnis wie die von Ihnen erstellte Klasse befinden. Alternativ zu den Methoden in `System.out` dürfen Sie auch die Methoden in `StdOut` benutzen.
Berechnen Sie die transponierte Matrix und geben Sie am Ende sowohl die eingegebene, als auch die transponierte Matrix aus.

Beispiel:

```
Matrix:     Ergebnis:
1 2 3       1 4 7
4 5 6       2 5 8
7 8 9       3 6 9
```

Hinweis: https://de.wikipedia.org/wiki/Matrix_(Mathematik)#Die_transponierte_Matrix