Übungsblatt 9

Betrügen Sie sich nicht selbst, indem Sie Programme zu den folgenden Aufgabenstellungen googeln und irgendwelches Zeug abschreiben, das Sie nicht verstehen.
In dieser Übung sollen Sie eine Klasse zum Umgang mit Matrizen schreiben. Zur Wiederholung daher hier nun ein paar Definitionen aus der Linearen Algebra.

(i) Ein n-dimensionaler *Vektor* über einem Körper K ist ein Tupel $(a_1, \ldots, a_n)$, so dass $a_i \in K$.

(ii) Sind $\alpha \stackrel{\text{def}}{=} (a_1, \ldots, a_n)$ und $\beta \stackrel{\text{def}}{=} (b_1, \ldots, b_n)$ Vektoren, so ist $\alpha + \beta = (a_1 + b_1, \ldots, a_n + b_n)$.

(iii) Ist $\alpha \stackrel{\text{def}}{=} (a_1, \ldots, a_n)$ ein Vektor über einem Körper K und ist $t \in K$, so ist $t \cdot \alpha = \alpha \cdot t = (t \cdot a_1, \ldots, t \cdot a_n)$.

(iv) Sind $\alpha \stackrel{\text{def}}{=} (a_1, \ldots, a_n)$ und $\beta \stackrel{\text{def}}{=} (b_1, \ldots, b_n)$ Vektoren, so sei $\alpha \cdot \beta \stackrel{\text{def}}{=} \sum_{i=1}^{n} a_i b_i$.

(v) Eine $m \times n$-Matrix A über einem Körper K hat die Form

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}$$

wobei alle $a_{ij} \in K$ sind.

(vi) Für eine Matrix A über einem Körper K (wie in (v)), sowie einem weiteren Körperelement $t \in K$ gilt:

$$t \cdot A = A \cdot t = \begin{pmatrix} t \cdot a_{11} & t \cdot a_{12} & \ldots & t \cdot a_{1n} \\ t \cdot a_{21} & t \cdot a_{22} & \ldots & t \cdot a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ t \cdot a_{m1} & t \cdot a_{m2} & \ldots & t \cdot a_{mn} \end{pmatrix}$$

(vii) Ist $\beta \stackrel{\text{def}}{=} (b_1, \ldots, b_n)$ ein n-dimensionaler Vektor über einem Körper K und ist A eine $m \times n$-Matrix über K (wie in (v)), so ist $A \cdot \beta = \gamma$ für den m-dimensionalen Vektor $\gamma = (c_1, \ldots, c_m)$ mit $c_i = \sum_{j=1}^{n} a_{ij} b_j$ für alle $i \in \{1, \ldots, m\}$.

(viii) Ist A eine $m \times n$-Matrix über einem Körper K (wie in (v)), und zudem B eine $n \times \ell$-Matrix über K der Form

$$B = \begin{pmatrix} b_{11} & b_{12} & \ldots & b_{1\ell} \\ b_{21} & b_{22} & \ldots & b_{2\ell} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n1} & b_{n2} & \ldots & b_{n\ell} \end{pmatrix}$$

so ist $A \cdot B = C$ für die $m \times \ell$-Matrix C der Form

$$C = \begin{pmatrix} c_{11} & c_{12} & \ldots & c_{1\ell} \\ c_{21} & c_{22} & \ldots & c_{2\ell} \\ \vdots & \vdots & \vdots & \vdots \\ c_{m1} & c_{m2} & \ldots & c_{m\ell} \end{pmatrix}$$

wobei $c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$ für alle $i \in \{1, \ldots, m\}$ und $j \in \{1, \ldots, \ell\}$ gilt.

(ix) Sind A und D je $m \times n$-Matrizen, wobei A wie in (v) definiert sei und die Einträge von D entsprechend durch d_{ij}. Dann ist

$$A + D = \begin{pmatrix} a_{11} + d_{11} & a_{12} + d_{12} & \dots & a_{1n} + d_{1n} \\ a_{21} + d_{21} & a_{22} + d_{22} & \dots & a_{2n} + d_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} + d_{m1} & a_{m2} + d_{m2} & \dots & a_{mn} + d_{mn} \end{pmatrix}$$

Im Moodle finden Sie eine Klasse `MathVec.h`, in der die Arithmetik für Vektoren umgesetzt ist. Ihre Aufgabe ist es, analog dazu eine Klasse für Matrizen zu schreiben.

Aufgaben

Schreiben Sie eine Template-Klasse `MathMatrix` mit den drei Template-Argumenten `T`, `rowDim` und `colDim`. Dabei ist `T` ein Typargument; jedes Element bzw. jeder Eintrag in die Matrix soll vom Typ `T` sein. Sie können davon ausgehen, dass für den Typ `T` die Operatoren $\star$ und + definiert sind.

Die Template-Argumente `rowDim` und `colDim` geben die Dimensionen der Matrix an, d.h. jeder Realisierung von `MathMatrix` sind über diese Parameter feste Dimensionen zugeordnet. Sie können davon ausgehen, dass beide Werte stets größer oder gleich 1 sind.

Alle zu implementierenden Funktionen sollen auf mögliche Fehler prüfen und im Falle eines solchen aussagekräftige Exceptions erzeugen.

Gehen Sie dabei in Ihrer Implementation Schritt für Schritt wie folgt vor:

1. Legen Sie ein Klassentemplate `MathMatrix` mit den oben genannten Template-Argumenten an. Fügen Sie der Klasse ein privates Attribut vom Typ `T**` hinzu, in dem die Elemente der Matrix gehalten werden sollen. Fügen Sie der Klasse ein weiteres Attribut `nullElement` vom Typ `T` hinzu. Letzteres brauchen wir um Nullmatrizen konstruieren zu können; nachdem wir `T` zu diesem Zeitpunkt noch nicht kennen, muss der Benutzer bei der Konstruktion der Matrix bestimmen, welches Objekt vom Typ `T` das Nullelement sein soll.

   Schreiben Sie einen Konstruktor mit dem Kopf

   ```
   MathMatrix(T x)
   ```

   der Speicher für `data` reserviert, das `nullElement` auf `x` setzt und alle Einträge der Matrix mit `nullElement` befüllt.

   Fügen Sie einen passenden Destruktor hinzu.

2. Schreiben Sie eine Methode mit dem Kopf

   ```
   T get(unsigned int row, unsigned int col) const
   ```

   die für eine Matrix A (wie in (v)) das Element a_{ij} zurückgibt, wobei $i =$ `row` und $j =$`col`.

   Schreiben Sie analog dazu eine Methode

   ```
   void set(unsigned int row, unsigned int col, T value)
   ```

   die den Wert eines Matrixeintrags setzt.

   Schreiben Sie außerdem eine Methode mit dem Kopf

   ```
   T getNullElement()
   ```

   die das `nullElement` zurückgibt.

3. Schreiben Sie eine Funktion mit dem Kopf

   ```
   string to_string(const MathMatrix<T, rowDim, colDim>& m)
   ```

die eine geeignete `string`-Repräsentation einer Matrix erzeugt. Sie können davon ausgehen, dass es auch eine `to_string`-Funktion für den Typ `T` gibt. Benutzen Sie diese Funktion im Folgenden, um zu überprüfen, dass Ihre Umsetzung der Aufgaben richtig funktioniert!

4. Schreiben Sie eine Methode mit dem Kopf

   ```
   MathVec<T, rowDim> getColumn(int index) const
   ```

   die die durch `index` identifizierte Spalte der Matrix in Form eines `MathVec` zurückgibt. Schreiben Sie analog dazu eine Methode `getRow`, die eine Zeile der Matrix zurückgibt.

5. Fügen Sie der Klasse einen Copy-Konstruktor hinzu und überladen Sie den Operator = als Copy-Wertzuweisung.

6. Fügen Sie der Klasse einen Move-Konstruktor hinzu und überladen Sie den Operator = als Move-Wertzuweisung.

7. Überladen Sie den Operator + im Kontext zweier kompatibler Matrizen gemäß der obigen Definition der Matrix-Addition.

8. Überladen Sie den Operator $\star$ im Kontext einer Matrix und einem dazu kompatiblen Körperelement gemäß der obigen Definition (vi) einer Matrix mit einem einzelnen Element.

9. Überladen Sie den Operator $\star$ im Kontext einer Matrix und einem dazu kompatiblen Vektor gemäß der obigen Definition (vii) einer Matrix mit einem Vektor.

10. Überladen Sie den Operator $\star$ im Kontext zweier kompatibler Matrizen gemäß der obigen Definition (viii) der Matrix-Multiplikation.

11. Bonusaufgabe

    Für einen Winkel δ ist durch

    $$R_\delta = \begin{pmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{pmatrix}$$

    eine zweidimensionale Rotationsmatrix gegeben. Das heißt: Ist v ein zweidimensionaler Vektor, so ist $R_\delta \cdot v$ derjenige Vektor, der entsteht, wenn man v um den Ursprung (d.h. um $(0,0)$) im Winkel δ rotiert.

    Benutzen Sie Ihre `MathMatrix`-Klasse, um ein Quadrat, das über dem Mittelpunkt eines `ViewPortGL` errichtet ist, um diesen Mittelpunkt rotieren zu lassen.

    Beachten Sie: Beim Koordinatensystem eines `ViewPortGL` befindet sich der Ursprung links oben, sein Mittelpunkt hat also die Koordinaten $(w/2, h/2)$, wenn der `ViewPortGL` die Breite w und die Höhe h besitzt.