1 Klasse MatrixVector

Die Klasse MatrixVector ist die Basisklasse aller weiteren Matrix – Klassen. Sie stellt im wesentlichen Methoden zur Ein- und Ausgabe, sowie zum Abfragen von verschiedenen Eigenschaften von Matrizen zur Verfügung.

1.1 Attribute

Die Klasse MatrixVector enthält die folgenden Attribute (private):

Attribut	**Bedeutung**
elements	Ist vom Typ vector<vector<T>> Enthält alle Elemente der Matrix
rows	Entspricht der Zeilenanzahl der Matrix
columns	Entspricht der Spaltenanzahl der Matrix

1.2 Methoden

Die Klasse MatrixVector enthält die folgenden Methoden:

Es sind vier Konstruktoren:

- Zur Erstellung einer Matrix ohne Übergabe von Parametern
  (es wird automatisch eine 3x3 Matrix erstellt)
- Zur Erstellung einer Matrix mit Übergabe eines Parameters
  (es wird eine quadratische Matrix entsprechend des übergebenen Parameters erstellt)
- Zur Erstellung einer Matrix mit Übergabe von zwei Parametern
  (es wird eine Matrix entsprechend der übergebenen Parameter (Zeilenanzahl, Spaltenanzahl) erstellt)

(jeweils Initialisierung aller Elemente mit Null!)

- Copy Konstruktor

sowie ein „virtueller“ Destruktor definiert worden.

Methode	**Bedeutung**
init()	Initialisierungsroutine für Copy-Konstruktor
resize()	Initialisierung der Matrix
getRows()	Zeilenanzahl zurückgeben
getColumns()	Spaltenanzahl zurückgeben
size()	Größe der Matrix in Form rows x columns zurückgeben
setRows()	Zeilenanzahl neu definieren (Achtung!!! Elemente der Matrix werden auf 0 initialisiert)
setColumns()	Spaltenanzahl neu definieren (Achtung!!! Elemente der Matrix werden auf 0 initialisiert)
operator <<()	shift-operator << überladen für die Ausgabe der Matrix
operator >>()	shift-Operator >> überladen zur Eingabe einer Matrix
operator []()	Indexoperator überladen zum Zugriff auf einzelne Zeilen der Matrix Gekapselt, da es sonst möglich wäre einzelne Elemente, Zeilen oder Spalten mithilfe der Memberfunktion, der Klasse vector zu zerstören

operator ()()	Indexoperator überladen zum Zugriff auf ein einzelnes Element der Matrix, dabei ist der row = 1 auch Zeile 1 der Matrix, sowie column = 1 auch Spalte 1 der Matrix
operator ==()	Überladen des Vergleichsoperators ==
operator !=()	Überladen des (Vergleichsoperators) !=
getRowElements()	Gibt alle Elemente einer bestimmten Zeile als Zeilenvektor zurück
getColumnElements()	Gibt alle Elemente einer bestimmten Spalte als Spaltenvektor zurück
isSquare()	Prüft ob es sich um eine quadratische Matrix handelt
isZero()	Prüft ob es sich um eine Nullmatrixhandelt und gibt entsprechend true oder false zurück
isUnit()	Prüft ob es sich bei der Matrix um eine Einheitsmatrix handelt.
isDiagonal()	Prüft ob es sich um eine Diagonalmatrix handelt
isScalar()	Prüft ob es sich um eine Skalarmatrix handelt
isUpper()	Prüft ob es sich um eine rechte/obere Dreiecksmatrix handelt.
isLower()	Prüft ob es sich um einer linke/untere Dreiecksmatrix
isTriangular()	Prüft auf Dreiecksmatrix und gibt entsprechenden Zahlenwert zurück bei oberer und unterer Dreiecksmatrix (Diagonalmatrix) wird eine 3 zurückgegeben bei oberer Dreiecksmatrix wird 1 zurückgegeben bei unterer Dreiecksmatrix wird 2 zurückgegeben handelt es sich nicht um eine Dreiecksmatrix wird eine 0 zurückgegeben
isColumnVector()	Prüft ob es sich um einen Spaltenvektor handelt
isRowVector()	Prüft ob es sich um einen Zeilenvektor handelt
swapRow()	Tauscht Zeilen aus
swapColumn()	Tauscht Spalten aus
operator =()	Zuweisungsoperator

1.3 Ausnahmebehandlungen

Die Klasse MatrixVector greift bei den von dieser Klasse ausgelösten Ausnahmebehandlungen auf die Klasse MatrixException zurück. Dabei sind die Fehlernummern 100 bis 199 für die Klasse MatrixVector vorgesehen. Derzeit sind folgende Ausnahmen deklariert:

Fehlernummer	Fehlermeldung	auslösende Methode
100	`Die Angabe der Zeilenanzahl und/oder Spaltenanzahl ist kleiner als 1.`	
101	`Sie haben eine Zeilenanzahl kleiner 1 eingegeben.`	
102	`Sie haben eine Spaltenanzahl kleiner 1 eingegeben.`	
103	`Zeilen- und/oder Spaltenangabe lag ausserhalb des Definitionsbereiches der Matrix.`	
104	`Die Zeilen- und Spaltenanzahl der Matrizen stimmen nicht ueberein.`	
105	`Die Zeilenangabe lag ausserhalb des Definitionsbereiches der Matrix.`	
106	`Die Spaltenangabe lag ausserhalb des Definitionsbereiches der Matrix.`	

1.4 Test

1.4.1 Testfälle

Die folgenden Testfälle wurden mit der Programmversion „`Version 1.8.6`“ `für die Datentypen` *`float, double, complex`* `und` *`Bruch`* `durchgeführt.`

Testfälle			
Nr.	**Methode**	**Testszenario**	**Beschreibung**
1	MatrixVector()	$\begin{matrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{matrix}$	Erzeugen einer Matrix durch den Standardkonstruktor. Ohne Übergabe von Parametern, wird automatisch eine 3x3 Matrix erzeugt und mit Null initialisiert. mit resize(); init()
2	MatrixVector (dim)	$\begin{pmatrix}0 & 0\\0 & 0\end{pmatrix}$ für dim=2 $\begin{pmatrix}i1.1 & \cdots & i1.x\\ \vdots & \ddots & \vdots\\ ix.1 & \cdots & ix.x\end{pmatrix}$ Für dim=x=10,... Alle Elemente mit 0 initialisiert	Erzeugen einer Matrix mit Übergabe eines Parameters . Ausgabe und Initialisierung mit dem Wert Null. mit resize(); init()
3	MatrixVector (rows,columns)	$\begin{matrix}0 & 0\\0 & 0\\0 & 0\end{matrix}$ für (3,2) ebenfalls für (3,4);(6,4);(10;10);...	Erzeugen einer Matrix mit Übergabe von zwei Parametern. Ausgabe und Initialisierung mit dem Wert Null. mit resize(); init()
4	getRows()	$\begin{matrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{matrix}$	Rückgabe der Anzahl der Zeilen: 3
5		$\begin{matrix}0 & 0\\0 & 0\\0 & 0\end{matrix}$	Rückgabe der Anzahl der Zeilen: 3
6	getColumns()	$\begin{matrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{matrix}$	Rückgabe der Anzahl der Spalten: 3
7		$\begin{matrix}0 & 0\\0 & 0\\0 & 0\end{matrix}$	Rückgabe der Anzahl der Spalten: 2
8	size()	$\begin{matrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{matrix}$ Matrix 2x2 Matrix 10x10	Rückgabe der Anzahl der Elemente: 9 Rückgabe der Anzahl der Elemente: 4 Rückgabe der Anzahl der Elemente: 100

<table>
<tr><td>9</td><td>setRows()</td><td>0 0 0 0
0 0 0 0
0 0 0 0 ->
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0</td><td>Aus einer Matrix (6x4) wurde eine Matrix (5x4) erzeugt.
Initialisierung der Elemente mit Null.</td></tr>
<tr><td>10</td><td>setColumns()</td><td>0 0 0 0
0 0 0 0
0 0 0 0->
0 0 0 0
0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0</td><td>Aus einer Matrix (5x4) wurde eine Matrix (5x6) erzeugt.
Initialisierung der Elemente mit Null.</td></tr>
<tr><td rowspan="2">11</td><td rowspan="2">operator >>()
und
operator <<()</td><td>Eingabe für Matrixwerte:
1 2 3 4 5 6 7 8 9</td><td>Einfache Eingabe von Werten für eine Matrix</td></tr>
<tr><td colspan="2">Ausgabe:
1 2 3
4 5 6
7 8 9</td></tr>
<tr><td rowspan="2">12</td><td rowspan="2"></td><td>Eingabe für Matrixwerte
1 22 333 4444 55555
666666 7777777
88888888 999999999</td><td>Einfache Eingabe von Werten für eine Matrix</td></tr>
<tr><td colspan="2">Ausgabe:
1 22 333
4444 55555 666666
7.77778e + 006 8.88889e + 007 1e + 009</td></tr>
<tr><td rowspan="2">13</td><td rowspan="2"></td><td>Eingabe für Matrixwerte
1 2 3 4 d r t e 4</td><td>Einfache Eingabe von Werten für eine Matrix
(Zahlen und Buchstaben)</td></tr>
<tr><td colspan="2">Ausgabe:
1 2 3
4 0 0
0 0 0</td></tr>
<tr><td rowspan="2">14</td><td rowspan="2"></td><td>Prinzip:
m1 >> m2
m1 << m2
Eingabe für Matrixwerte
1 2 3 4 5 6 7 8</td><td>Mehrfache Ein- und Ausgabe von Matrizen.</td></tr>
<tr><td colspan="2">Ausgabe:</td></tr>
</table>

		1 2 3 4 5 6 7 8	
15	operator []()	0 0 0 0 0 0 0 0 0	Zugriff auf einzelne Zeilen der Matrix.
16	operator ()() operator =()	0 0 0 0 0 0->0 0 0 0 0 0 0 0 99 0 0 0 0	Zuweisung eines Wertes zu einem Bestimmten Element. hier: (2,2)=99
17	getColumnElements()	1 2 3 4 5 6->1 7 8 9 4 7	Gibt eine Spalte als Spaltenvektor zurück
18	getRowElements()	1 2 3 4 5 6->1 2 3 7 8 9	Gibt eine Zeile als Zeilenvektor zurück
19	operator ==()	0 0 0 0 == 0 0 0 0	Prüft zwei Matrizen auf Gleichheit Hier true.
		0 0 0 0 == 1 2 3 4	Prüft zwei Matrizen auf Gleichheit Hier false.
20	operator !=()	0 0 0 0 != 0 0 0 0	Prüft zwei Matrizen auf Ungleichheit Hier false.
		0 0 0 0 != 1 2 3 4	Prüft zwei Matrizen auf Ungleichheit Hier true.
21	isSquare()	0 0 0 0	Prüft, ob Matrix quadratisch ist. Hier true.
22		0 0 0 0 0 0	Prüft, ob Matrix quadratisch ist. Hier false.
23	isZero()	0 0 0 0	Prüft, ob Matrix eine Nullmatrix ist. Hier true.
24		1 2 3 4	Prüft, ob Matrix eine Nullmatrix ist. Hier false.
25	isUnit()	1 0 0 1	Prüft, ob Matrix eine Einheitsmatrix ist. Hier true.
26		1 5 9 3	Prüft, ob Matrix eine Einheitsmatrix ist. Hier false.
27	isDiagonal()	2 0 0 1	Prüft, ob Matrix eine Diagonalmatrix ist. Hier true.

28		1 1 1 1	Prüft, ob Matrix eine Diagonalmatrix ist. Hier false.
29	isScalar()	1	Prüft, ob (Matrix) ein Skalar ist. Hier true.
30		1 1 1 1	Prüft, ob (Matrix) ein Skalar ist. Hier false.
31	isUpper()	1 1 0 1	Prüft, ob eine rechte/obere Dreiecksmatrix vorliegt. Hier true.
32		1 1 1 1	Prüft, ob eine rechte/obere Dreiecksmatrix vorliegt. Hier false.
33	isLower()	1 0 1 1	Prüft, ob eine linke/ untere Dreiecksmatrix vorliegt. Hier true.
34		1 1 1 1	Prüft, ob eine linke/ untere Dreiecksmatrix vorliegt. Hier false.
35	isTriangular()	1 1 0 1	Gibt 1 zurück. Diese Matrix ist eine obere Dreiecksmatrix
36		1 0 1 1	Gibt 2 zurück. Diese Matrix ist eine untere Dreiecksmatrix
37		1 2 3 4 5 6 7 8 9	Gibt 0 zurück. Diese Matrix ist eine keine Dreiecksmatrix
38		1 0 0 0 2 0 0 0 1	Gibt 3 zurück. Diese Matrix ist eine obere und untere Dreiecksmatrix - Diagonalmatrix
39	isColumnVector()	0 0 0	Prüft, ob Matrix ein Spaltenvektor ist. Hier true.
40		0 0 0	Prüft, ob Matrix ein Spaltenvektor ist. Hier false.
41	isRowVector()	0 0 0	Prüft, ob Matrix ein Zeilenvektor ist. Hier true.
42		0 0 0	Prüft, ob Matrix ein Zeilenvektor ist. Hier false.
43	swapRow()	1 2 3 4 5 6-> 7 8 9 1 2 3 7 8 9 4 5 6	Tauscht zwei Zeilen der Matrix aus.
44	swapColumn()	1 2 3 4 5 6-> 7 8 9 2 1 3 5 4 6 8 7 9	Tauscht zwei Spalten der Matrix aus.
	Tests für Fehler		
45	Matrix mit der Dimension (0,1) anlegen Matrix<float> q(0,1);		Fehler 100 wird ausgelöst.

46	Zeilenanzahl kleiner Eins. p.setRows(0)	Fehler 101 wird ausgelöst.
47	Spaltenanzahl kleiner Eins. p.setColumns(0)	Fehler 102 wird ausgelöst.
48	Spalte ausserhalb der definierten Dimension p.swapColumn(1,4)	Fehler 103 wird ausgelöst.
49	Zuweisung Matrizen unterschiedlicher Dimensionen r=n	Fehler 104 wird ausgelöst.
50	Zeilenangabe ausserhalb des Definitionsbereiches der Matrix. p.getRowElements(100)	Fehler 105 wird ausgelöst.
51	Spaltenangabe ausserhalb des Definitionsbereiches der Matrix. p.getColumnElements(100)	Fehler 106 wird ausgelöst.

(Testfälle wurden für mehrere Testszenarien ausgeführt – siehe Testprogramm.)

1.4.2 Testprotokoll

Testfall	Bestanden	Bemerkung	Datum	Version
1	Ja		20.06.2010	1.8.6
2	Ja		20.06.2010	1.8.6
3	Ja		20.06.2010	1.8.6
4	Ja		20.06.2010	1.8.6
5	Ja		20.06.2010	1.8.6
6	Ja		20.06.2010	1.8.6
7	Ja		20.06.2010	1.8.6
8	Ja		20.06.2010	1.8.6
9	Ja		20.06.2010	1.8.6
10	Ja		20.06.2010	1.8.6
11	Ja		20.06.2010	1.8.6
12	Ja	Ausgabe: 1 22 333 4444 55555 666666 $7.77778e+006$ $8.88889e+007$ $1e+009$ Ab der siebten Stelle wird aufgerundet	20.06.2010	1.8.6
13	Ja	Ausgabe: 1 2 3 4 0 0 0 0 0 Ab der Eingabe eines Buchstabens wird die Eingabe abgebrochen und der Programmablauf setzt sich fort.	20.06.2010	1.8.6
14	Ja		20.06.2010	1.8.6
15	Ja		20.06.2010	1.8.6
16	Ja		20.06.2010	1.8.6
17	Ja		20.06.2010	1.8.6
18	Ja		20.06.2010	1.8.6
19	Ja		20.06.2010	1.8.6
20	Ja		20.06.2010	1.8.6
21	Ja		20.06.2010	1.8.6
22	Ja		20.06.2010	1.8.6

23	Ja		20.06.2010	1.8.6
24	Ja		20.06.2010	1.8.6
25	Ja		20.06.2010	1.8.6
26	Ja		20.06.2010	1.8.6
27	Ja		20.06.2010	1.8.6
28	Ja		20.06.2010	1.8.6
29	Ja		20.06.2010	1.8.6
30	Ja		20.06.2010	1.8.6
31	Ja		20.06.2010	1.8.6
32	Ja		20.06.2010	1.8.6
33	Ja		20.06.2010	1.8.6
34	Ja		20.06.2010	1.8.6
35	Ja		20.06.2010	1.8.6
36	Ja		20.06.2010	1.8.6
37	Ja		20.06.2010	1.8.6
38	Ja		20.06.2010	1.8.6
39	Ja		20.06.2010	1.8.6
40	Ja		20.06.2010	1.8.6
41	Ja		20.06.2010	1.8.6
42	Ja		20.06.2010	1.8.6
43	Ja		20.06.2010	1.8.6
44	Ja		20.06.2010	1.8.6
45	Ja		20.06.2010	1.8.6
46	Ja		20.06.2010	1.8.6
47	Ja		20.06.2010	1.8.6
48	Ja		20.06.2010	1.8.6
49	Ja		20.06.2010	1.8.6
50	Ja		20.06.2010	1.8.6
51	Ja		20.06.2010	1.8.6