J-STEM: Matrix Manipulation Language

COMS W4115

February 8, 2017

Manager: Julia Troxell (jst2152)
Language Guru: Tessa Hurr (trh2124) & Emily Song (eks2138)
Tester: Michelle Lu (ml3720)
System Architects: Samantha Stultz (sks2200)

1. Description

J-STEM is a matrix manipulation language. Matrices are important tools in mathematics with various applications such as graphic rendering and encryption. We have found that matrix manipulation in Java is tedious and difficult. We propose J-STEM, a language that allows for easy and intuitive matrix transformations.

2. Purpose

The matrix manipulation language will have functions to transform matrices. The functions will include calculating the determinant, rotating the matrix, applying operations to each value in the matrix, and deleting or adding columns and rows. Basic functions to add, subtract, multiply, and divide matrices will be implemented as well. The goal of our language is to utilize these functions and apply them to images. The language will be able to edit various features of images. These include different color overlays, image rotations, and altering contrast and brightness.

3. Syntax

Comments

//	single-line comment
/* … */	multi-line comment

Data Type	Description	Declaration
mx	A matrix (a list of rows) Default initialization to a 0x0 matrix (an empty list)	mx m = matrix(); mx m = matrix(1, 2);
row	A list	row r = [1, 2, 3];
string	String value	string str = “hello!”;
int	Integer value	int x = 5;
float	Floating point value	float x = 5.0;
column	A list	col c = [1, 2, 3];
equation	A tuple, where <1,2,3> represents the equation 1 + 2x + 3x^2	eq e = <1,2,3>;
file	file	file filename = file.ppm;
list	Like Java	list l[];

Keyword	Description	Usage
print()	Prints to console	print(m);
for	Iteration until condition is reached	for(condition) { … }
while	Loop until condition is reached	while(condition) { … }
range	Range of numbers used in a for loop	range(0, 9, 2) *// 0, 2, 4, 6, 8*
if	If in an if-else statement	if { … }
else	Else in an if-else statement	else { … }
else if	Additional if statement after an if-else statement	else if { … }
return	Returns value from a function	return m;

main	Main function	def main:

addRow(int rowNum, row r)	Insert a row of 0's at the bottom of the matrix (end of list)
	If optional arg1 is supplied, insert a row of 0's at that index
	If arg1 and arg2 are supplied, insert arg2 (a row) and index arg1
		m.addRow();
		m.addRow(2);
		m.addRow(2, [1,2,3]);

delRow(int rowNum)	Delete last row in matrix (end of list) If optional argument supplied, delete row at that index	m.delRow(); m.delRow(2);
matrix(int x, int y)	Create a matrix Arguments specify #rows, #columns No arguments creates a 0x0 matrix	mx m = matrix(); mx m = matrix(2, 3);
applyEq(eq e)	Applies equation to each matrix cell	m.applyEq(eq e);
loadFile(file filename)	Load an image file into a matrix	mx m = loadFile(file.ppm);
transpose()	Transpose matrix	m.transpose();

addColumn(int colNum, row r)	Insert a column at the right of the matrix
	If optional arg1 is supplied, insert a column of 0's at that index
	If arg1 and arg2 are supplied, insert arg2 (a column) and index arg1
		m.addColumn();
		m.addColumn(3);
		m.addColumn([1,2,3]);

delColumn(int colNum)	Delete rightmost column in matrix If optional argument is supplied, delete column at that index	m.delColumn(); m.delColumn(3);

findDet()	Calculates and returns the determinant of a matrix.	int det = matrix.findDet();
inverse()	Takes the inverse of the matrix	m.inverse();
colorize(string color)	Colorizes the image based on the supplied argument (‘red’, ‘green’, ‘blue’, ‘b&w’)	m.colorize(‘b&w’);
contrast(string direction), brightness(string direction), saturation(string direction)	Increase or decrease the contrast, brightness, or saturation	m.contrast(‘increase’); m.brightness(‘decrease’); m.saturation(‘increase’);
height(mx m)	Returns # of rows in matrix	int h = height(m)
width(mx m)	Returns # of columns in matrix	int w = width(m)
elements(eq e)	Returns # of elements in equation	int el = elements(e)

Operator	**Description**	**Usage**
=	Assignment operator	int y = 6, int z = 2
+	Arithmetic operators	int x = y + z *// x = 8*
-	Subtraction operator	int x = y - z *// x = 4*
*	Multiplication operator	int x = y * z *// x = 12*
/	Division operator	int x = y / z *// x = 3*
**	Exponentiation operator	int x = 2 ** 3 *// x = 8*
==	Returns 1 if the values are equal, 0 otherwise	y == z *// return 0*
+=	Adds the value on the left to the value on right and stores in left variable	y += 1 *// y = 7*
++	Increment operator	z++ *// z = 3*

--	Decrement operator	z-- // *z = 1*
!=	Returns 1 if the values are not equal, 0 otherwise	y != z // *return 1*
>	Greater than operator	y > z // *return 1*
<	Less than operator	y < z // *return 0*
>=	Greater than or equal to operator	y >= z // *return 1*
<=	Less than or equal to operator	y <= z // *return 0*
&&	Logical AND operator	0 && 1 // *return 0*
||	Logical OR operator	0 && 1 // *return 1*
!	Logical NOT operator	!(5 == 5) // *return 0*
e<i>	Access index i of the equation	eq e = <1,2,3>; e<1> // *2x*

Matrix Operator	**Description**	**Usage**
+, -, *, /, =	Scalar and matrix operations	M = 5 * M_1 // *scalar multiplication* M = M_1 * M_2 // *matrix multiplication*
==, !=, >, <, >=, <=	If two matrices have the same dimension: Compare element in one matrix to corresponding element in other matrix, for entire matrix	M1 == M2 // *returns 1 if matrices contain same elements, 1 otherwise*
M[r, :]	Access row r of matrix M	M[2, :] // *access row 2*
M[:, c]	Access column c of matrix M	M[:, 4] // *access column 4*

4. Examples

```
// colorize an image file!

def colorize(string color):
      eq e;
      if color == ‘red’:
            e = <1,2,3>; // arbitrary, we will look up real equations later
      else if color == ‘green’:
            e = <2,3,4>;
      else if color == ‘blue’:
            e = <3,4,5>;
      else if color == ‘b&w’:
            e = <4,5,6>;
      m.applyEq(e);
      return;

def applyEq(eq e):
      for i in range(height(m)):
            for j in range(width(m)):
                  int x = m[i,j];
                  for k in range(elements(e)):
                        x += e<k> * x**k
                  m[i,j] = x;
      return;

def main:
      mx m = loadFile(filename); // we will handle the I/O specifics later
      m.colorize(‘red’);
      return m;
```