M^2 Language Proposal

Tengyu Zhou - tz2338
Jeffrey Monahan - jm4155
Chrisitne Pape - cmp2223
Shelley Zhong - sz2699

September 26, 2017

1 Motivation

Matrices are an integral part of many mathematical operations and have uses ranging anywhere from graphics to cryptography. Some of the operations, however, that can be performed on them can get messy and complicated very quickly when done by hand. By creating this language, we hope to optimize how people work with matrices, saving time both on initial computation and going back to fix potential errors.

2 Language Description

The primary goal of our language is to provide a platform to both easily define and manipulate matrices. To this end, we aim to implement the basic building blocks of all complex matrix functions (e.g. cross product, inverse, transpose, etc.) as well as implement an intuitive method to define and manipulate vectors of various sizes. Furthermore, we will implement the fundamental operators needed to perform most mathematical operations. We feel that it is important the program has the ability to interact with the user through I/O, so we keep the idea of strings, printing, and scanning user input. We also keep basic fundamentals, such as conditional statements and loops. Our syntactical style while remain similar to the widely used languages (i.e. We will be keeping our language easy enough to learn by someone who already knows modern programming languages while adding enough Matrix-specific features so that it is specialized.

Ultimately, our programming language is designed to be used as a tool for manipulating matrices, so the types of programs meant to be written in this language reflect that. Besides performing basic operations, programmers should be able to build programs that can solve systems of equations, encrypt/decrypt messages, or even create graphs and images.

3 Syntax

3.1 Reserved Words

if	else	and	or	return	while	for	break	continue	to

3.2 Primitive Data Types

int	Standard representation of signed or unsigned integers. Example declaration: int X = 1;
float	Standard representation of signed or unsigned single-precision floating point numbers. Example declaration: float X = 1.0;
double	Double-precision floating point number. Example declaration: double X = 1.0;
char	16 bit datatype representation. Example declaration: char C = 'a';
boolean	Standard representation (holds values 1 or true, for true, and 0 or false, for false. Example declaration: bool X = false;

3.3 Support Data Types

matrix	Simple definition: similar to the definition of an array in Java (e.g. a 2 × 3 matrix is defined by the user as Matrix M = new Matrix[2, 3]) Storage type: stored as an array of the elements, where the top left is the 0th element and the bottom right is the last.
string	Strings are char arrays. Example declaration: string A = new string("hello world")
array	Instantiated as in Java Example declaration: int[] A = new int[10];
graph	Stored as screen vertex data. Each vertex is either filled or not and is filled based upon
function	For ease of input when doing mathematical operations. Example declaration: function Y = $3x + 5y$

3.4 Basic Operators

$+, -, *, \times, /, \%$	Arithmetic operators
$==, ! =, >, <, >=, <=$	Relational operators
!, &&, ||	Logical operators
$=$	Assignment operator

3.5 Matrix Operators

.	Dot product A.B
×	Cross product A × B
$[x : y : z]$	Matrix construction operator
det(A)	Compute the determinant of a matrix
transpose(A)	Compute the transpose of a matrix
makeMatrix(list,x,y)	Return a matrix from an array with specified dimensions
.length	Returns number of columns
.width	Returns number of rows

3.6 Graphs functions

Graph A = new Graph[function];	Fills in screen vertex data so that the function may be displayed on screen. The filled vertices can be manipulated using matrices
writeGraph(A);	Writes Graph A to the screen and displays the result in a window.

Graph A; Matrix B;
Graph C = A * B;
	Reorders filled status of vertices in Graph A, according to Matrix B, in order to produce the new Graph C. This is, essentially, matrix-vector multiplication done iteratively.

3.7 General Syntax

(x), function(x)	Parentheses to order calculation process as well as enact functions (i.e. det(A), transpose(A), etc.)
;	Denotes end of line (white space is ignored)
{statements}	Curly braces are used to group statements
VARIABLE_NAME	Identifier variables will be all uppercase with words separated by underscores
functionvariables	Variables used in functions will be all lowercase
//comments	For comments
\”, \ ’, \t, \n, \\	Special characters: ”, ’, tab, newline, \

3.8 Other Built-in Functions

readInput()	Get user input from the keyboard
print()	Print content in the parenthesis
import()	Import libraries
exit()	Exit the program

4 Sample Code

[1]

```
//takes a message and a key from the user
//encrypts the message and prints encryption
//decrypts the message and prints the message

char[] LETTER_DEFINITION = {'','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};

String MESSAGE = readInput();
int[] INT_MESSAGE = new int[MESSAGE.length]

for(int I = 0; I<MESSAGE.length; I++){
  bool CHANGED = false;
  for(int J = 0; J<LETTER_DEFINITION.length; J++){
    if(MESSAGE[I] ==LETTER_DEFINITION[J]){
      INT_MESSAGE[I] = J;
      CHANGED = true;
    }
  }
  if(!CHANGED){
    print(”Your message contains an invalid character. Please use only letters
   and spaces.”);
    exit(1);
  }
}

int KEY_LENGTH = readInput();
int[] KEY = new int[KEY_LENGTH];

for(int I = 0; I < KEY_LENGTH; I++){
  KEY[I] = readInput();
}

matrix KEY_MATRIX = makeMatrix(KEY, 3, 3);
matrix MESSAGE_MATRIX = makeMatrix(INT_MESSAGE, len(INT_MESSAGE)/3, 3);

matrix CIPHER_TEST_MATRIX = KEY_MATRIX * MESSAGE_MATRIX;

for(int I = 0; I<CIPHER_TEST_MATRIX.length; I++){
  for(int J = 0; J<CIPHER_TEST_MATRIX.width; J++){
    //print coded message
    print(CIPHER_TEST_MATRIX[I][J] + ””);
  }
}
print(   \n   );

//start decrypt
Matrix D_KEY = KEYˆ-1;

Matrix D_MESSAGE = D_KEY * CIPHER_TEST_MATRIX;

for(int I = 0; I<D_MESSAGE.length; I++){
  for(int J = 0; J<D_MESSAGE.width; J++){
    //print decoded message
    int INDEX = D_MESSAGE[I][J];
    print(LETTER_DEFINITION[INDEX]);
  }
}
```

References

[1] Application to cryptography, http://aix1.uottawa.ca/ jkhoury/cryptography.htm.