The MatriCs

Reloaded

A linear algebra-specific language for the budding C enthusiast

Short name: MaC
Extension: .neo

Talal Asem Toukan [tat2132] - Manager
Emmanuel Koumandakis [ek2808] - System Architect
Duru Kahyaoğlu [dk2565] - Language Guru
Florian Shabanaj [fs2564] - Language Guru
Nikhil Raghav Baradwaj [nrb2129] - Tester

Motivation:

Matrices are a useful tool to represent finite sets of data across a wide range of subject matter, including the hard sciences, mathematics, engineering, and computer science. Moreover, basic matrix operations, including matrix multiplication, summation, subtraction, inversion, and finding the transpose, eigenvalues, and eigenvectors of matrices, have been unusually difficult to perform without the use of external packages in languages like Python and R. The purpose of our language, The MatriCs, is to not only simplify these computations, but also to reduce the running time of matrix operations. Using C-like syntax, our language is tailored for programmers familiar with C, but not necessarily familiar with popular “data science” packages, like SciPy (Python), NumPy (Python), and Matrix (R).

Description:

MatriCs is a strongly typed language that combines a C-like syntax with a whole host of special operators. These operators enable the user to perform fundamental computations involving linear algebra, including but not limited to calculating a matrix’s transpose and inverse, carrying out matrix multiplication, and extracting submatrices through slicing. At the very core of our language is the special data type: matrix. MatriCs will compile to LLVM.

Data Types:

Primitive Data Types

Type	*Description*	*Syntax*	*Example*
integer	Used to define integer types	int	int a = 5
float	32-bit floating number	float	float a = 5.0
bool	Used to define boolean types (true and false)	bool	bool flag = true
double	64-bit floating number	double	double d = 5.0

Special Data Types

Type	*Description*	*Syntax*	*Example*
matrix	Defines a matrix	mat	mat int matr_x = {1,2,3; 4,5,6; 7,8,9};

Operators:

Standard Operators

Name	*Syntax*	*Example*
Addition,Subtraction, Multiplication, Division	+,-,*,/	int a = 5 + 8 //a = 13

Addition, Subtraction, Multiplication, Division Assignment	+=, -=, *=, /=	int a = 4;
		a += 2;
		//a = 6

Assignment	=	int a = 7 //a has a value of 7
Equality check	==	7 == 7 //Returns 1

Greater than	>	6 > 5 //Returns 1
Less than	<	5 < 3 //Returns 0
Greater than or equal to	=>	5 => 4 //Returns 1
Less than or equal to	<=	5 <= 5 //Returns 1
Not equal	!=	5 != 3 // Returns 1

Logical Not	!	int a = 1;
		int b = 0;
		!(a && b) //Returns 1

Logical AND	&&	//with the values from the example above a && b //Returns 0
Logical OR	||	//with the values from the example above a || b //Returns 1

Matrix Operators

Name	*Description*	*Syntax*	*Example*

Scalar Multiplication, Scalar Division, Scalar Power	Element-wise multiplication/ division or scalar multiplication/ division/power	.*, ./, .^	mat int C = A.*B;
			mat int C = A./B;
			mat int C = A.*2;
			mat int C = A./2;
			mat int C = A.^2;

Matrix Multiplication, Matrix Division	Matrix multiplication/ division.Operation is not commutative.
	If at least one input is scalar, then A*B is equivalent to A.*B and is commutative.
		*, /	mat int C = A*B;
			mat int C = A/B;
			mat int C = A*3;
			mat int C = A/3;

Addition, Subtraction	Addition/ subtraction, scalar or	+, -	mat int C = A+B; mat int C = A+2;

	element-wise		mat int C = A+B; mat int C = A-2;
Transpose	Returns the transpose of a matrix	’	mat int B = A’;
Indexing	Returns the element in the specified row and column of a given matrix	matr_x[*row_index*][*column_index*]	matr_x[2][4] //returns the element in the 2nd row and 4th column of the given matrix
Slicing	Returns an array of elements with the specified location in terms of rows and columns	matr_x[row_index1:row_index2, column_index1:column_index2]	matr_x[1:2,2:4] //returns the matrix in the 1 to 2nd rows and 2 to 4th columns

Keywords:

Syntax	*Description*
if	Similar to C conditional
elif	Similar to C conditional (but with `elif` keyword instead of `else if`)
else	Similar to C conditional
for	Similar to C for loop
while	Similar to C while loop
{}	scope
return	Return from function
null	No data
void	Returns nothing
//	Inline comment
/* */	Block comment
;	End of instruction

Code Examples

```
main() {
  // note that indexing starts from 0 similar to
  // other conventional coding languages.
  mat int matr_x = {1,2,3; 4,5,6; 7,8,9;};
  matr_x[1][2]; // returns 6

  // close braces to indicate the end of main
}

int determinant_of_2x2(int mat a) {
  return a[0][0]*a[1][1] - a[1][0]*a[0][1];
}

int determinant(mat int m){
    int det = 0;
    int sign = 1;

    if(rows(m) == 1) {    // base case, dimensional array
      return m[0][0];
    }

    // lib functions to get # of rows and columns in m
    // (actually the values should be the same anyway)
    int rows = rows(m);
    int cols = cols(m);

    // finds determinant using row-by-row expansion
    for(int i = 0; i < rows; i++){

      // keep decomposing the matrix by 1 dimension
      mat int smaller_m[rows-1][cols-1];

      for(int a = 1; a < rows; a++){
        for(int b = 0; b < cols; b++){

          if(b < i){
            smaller_m[a-1][b] = m[a][b];
          }
          elif(b > i){
            smaller_m[a-1][b-1] = m[a][b];
          }
```

```
        }
      }

      if (i%2 == 0){    // sign changes based on i
        sign = 1;
      }
      else{
        sign = -1;
      }
      det += sign*m[0][i]*determinant(smaller_m); // rec call for det
    }
    return det;
}
```