MATVEC:

MATRIX-VECTOR COMPUTATION LANGUAGE REFERENCE MANUAL

John C. Murphy
jcm2105
Programming Languages and Translators
Professor Stephen Edwards

Language Reference Manual

Introduction

The purpose of this manual is to serve as a reliable reference manual for describing the MatVec language. This manual provides a general outline of the language and a detailed description of its grammar. The language describes the lexical conventions used by the language as well as the syntax notation, grammar, identifiers, and operators.

Lexical Conventions

Tokens

There are five classes of tokens that are used with this language. These tokens include: identifiers, keywords, constants, operators, and other separators. For the most part, white space is used to separate different tokens in the language. In the context of this language, white space refers to spaces, tabs, and new lines.

Comments

Comments are denoted by the use of the /* and */ characters (as in the C language). The /* character denotes the start of the comment and the */ character denotes the end of the comment string. The language does not support the use of nested comments. Single-line comments can be used by using the // symbol.

Identifiers

An identifier is a sequence of letters and numbers that represents a value. Like in the C language, the first character of an identifier must be a letter. The underscore is not supported as a letter in this language.

Keywords

The following identifiers are reserved as keywords in the MatVec language and may not be used for any other purpose:

const
else
endif
float
if
int
matrix
normalize
print
read
then
while
vector

Constants

A constant represents a fixed value that should be properly declared before it is used. There are four kinds of constants that are allowed in MatVec: an integer constant, a floating point constant, a vector constant, and a matrix constant.

These different types of constants correspond to the identifier types that are described in the Identifier Types section.

Meaning of Identifiers

Identifier Types

There are four basic identifier types in MatVec. The four types of identifiers that are allowed in MatVec are: integer type, floating point type, vector, and matrix. All of these types are described below:

An **integer** type is a sequence of digits that represents an integer. Only decimal (Base 10) numbers are supported in this language.

A **floating point** type is a sequence of digits that contains a decimal point somewhere in the value. In other words, a floating point number contains an integer part, a decimal point, and a fractional part. However, the floating point constant cannot consist of just a decimal point.

A **vector** type is either a mathematical entity that represents a vector.

A **matrix** type is a set of numbers that represents a matrix entity in mathematics.

Expressions

Primary Expressions

A Primary Expression consists of identifiers, constants, or expressions that are contained within parentheses. The parentheses around the expression do not affect the outcome of the contained expression.

Unary Plus Operator

The unary + operator is simply the result of the value of the operand. The operand can be either an integer or floating point number.

Unary Minus Operator

The unary – operator is the negative result of the value of the operand. The operand can be either an integer or floating point number.

Multiplicative Operators

The multiplicative operators of the language are * and /. Both of these operators require two operands and both operations are performed from left-to-right (left-associative operation). The * operator denotes multiplication and the / operator denotes division.

Additive Operators
The additive operators of the language are + and -. Both of these operators require two operands and the both operations are performed from left-to-right (left-associative operation). The + operator denotes multiplication and the / operator denotes division.

Relational Operators
The relational operators in the MatVec language are a subset of the relational operators that are used in the C language. Relational operators are used to determine whether a comparison between two operands is true or false and returns the result. Therefore, although the operation is left-associative, this fact by itself is not very useful. Since the expression a<b<c is parsed as (a<b)<c, this expression is invalid since (a<b) yields either true or false. The values of true or false cannot be one side of the relational operator.

These are the relational operators that are used by the MatVec language:
<, <=, >=, >

Less Than (<) operator
Less Than or Equal To (<=) operator
Greater Than (>) operator
Greater Than or Equal To (>=) operator

Equality Operators
The equality operators in the MatVec language are similar to the equality operators that are used in the C language. The equality operators are similar to the relational operators in that they will return a value of true or false.

These are the equality operators that are used by the MatVec language:
==, !=

Equal To (==) operator
Not Equal To (!=) operator

Assignment Expression
The assignment expression that is used in MatVec is the = operator. In the assignment expression, the value of the left operand of this expression, known as the lvalue, is reassigned to the result of the right operand. The lvalue of the expression, therefore, needs to be a valid identifier; it cannot be a constant.

Operator Precedence and Order of Evaluation
The following table shows the precedence and order of evaluation of the different operators used in MatVec. The order of precedence shown in the table is a subset of the Operator Precedence and Associativity hierarchy from the C language.

Operator Name	Operator Symbol	Operator Associativity
Unary Operator	+ -	Right to Left
Multiplicative Operator	* /	Left to Right
Additive Operator	+ -	Left to Right
Relational/Equality Operators	< <= >= > == !=	Left to Right
Assignment Operator	=	Right to Left

Declarations
In MatVec, identifiers and constants need to be declared before they are used. Declarations start with the identifier type followed by a list of identifiers or constants to be declared.

Here is an example of identifier declarations:
int i, j;
matrix u, v;
float g, h;

Here is an example of constant declarations:
const int x = 3, y = 5;
const matrix m = [1 2 3 | 4 5 6 | 7 8 9];

Statements
Statements are expressions in MatVec that are executed in sequential order. Statements in MatVec are similar to statements in C.

Conditional Statement
A conditional statement is an if-else statement where the code in the brackets is executed only when the "if" conditional is met. The "else" part of the conditional statement is optional and is only executed if the conditional is not met. Here is the iterative statement:

```
if (conditional) {
        statement;
        statement;
}
else {
        statement;
        statement;
}
endif
```

Iterative Statement
An iterative statement is the while loop that keeps executing a portion of code until the condition is satisfied. Here is the iterative statement:

```
while (conditional)
{
        statement;
        statement;
}
```

Functions
The following are functions that are available in MatVec:
read();
print();
normalize();

The print() statement can be used to print out an identifier of any type in the language. The print statement can print out results such as the output of an integer, a vector, or a matrix value.

The read() statement is used to read in an identifier value from standard input and assign the input to an identifier.

The normalize() function takes a vector as its input and computes the “normalized” values of the input vector.