Operators

Every variable or constant is described by its own value and belongs to a data type. Using operators and round brackets those can be applied to make expressions that actually are rules for obtaining new values. In the general case the expression consists of several elements (operands) and operation signs (operators), while the type of its value is defined via the type of operands and the type of operations applied to them.

If operations have equal priority those will be executed in sequence, from left to right. If priority of operation is higher, it will be executed first irrespective of position of operation sign. Also expressions embraced in brackets are executed first, after which those are considered as operands. In expression (x>0)**and**(y>0) brackets are necessary since relation operations have the lowest priority.

Unary operators:

|  |  |  |
| --- | --- | --- |
| Operator | Semantics | Purpose and example |
| - | - <expression> | Expression complementary operator  **Example:**  -1 |
| not | not <logical or integer-value expression> | Logical or integer-value expression operator.  **Example:**  **not** (A > 10) |
| + | + <expression> | Unary plus – to be ignored. |
| @ | @<variable> | It calculates the reference of variable data. Reference is 32-bit integer number corresponding to the variable address in the memory. |

Mathematical operators.

Real, complex and integer numbers, arrays, matrixes, as well as expressions of related types, can be used as operands in arithmetical operations. Operations with matrixes are defined according to linear algebra rules, while operations with arrays are executed elementwisely (in the last case sizes of arrays shall be the same). Massive is perceived as a row-vector. Column-vector is set as a matrix with the size of n×1, for example A=[[1],[3],[2]]; otherwise – as a transposed array: A=**transp**([1,3,2]);

|  |  |  |
| --- | --- | --- |
| Operator | Semantics | Purpose and example |
| **+** | <addend 1>+<addend 2> | Addition of two expressions. Addition corresponds to concatenation for strings. Matrixes and vectors are added elementwisely.  **Example:**  1 + 10; |
| **-** | <minuend>-<subtrahend> | Subtraction of the second expression from the first one. It is not defined for strings. Matrixes and vectors are subtracted elementwisely.  **Example:**  10 + 5  [1,1] + [5,7] |
| **\*** | <multiplier 1>\*<multiplier 2> | Multiplication of two expressions. It is not defined for strings. Vectors are multiplied elementwisely. Matrixes are multiplied according to matrix multiplication rules.  **Example:**  A \* B |
| **/** | <dividend>/<divider> | Direct division of the first expression by the second one. Vectors are divided elementwisely. For matrixes multiplication by an inverse matrix is performed.  **Example:**  3/10  [1,1]/[10,20] |
| **\** | <matrix A>\<matrix B> | Inverse division of matrixes. Corresponds to operation A-1\*B |
| **^** | <basic value>^<power> | Raising the first expression to the power of the second one. Vectors are processed elementwisely. Matrixes are raised to an integral power according to matrix multiplication rules. To raise a matrix to a power elementwisely use operator **.^**  **Example:**  2^3 |
| **.\*** | <matrix 1>.\*<matrix 2> | Elementwise multiplication of real or complex matrixes. |
| **./** | <matrix 1>./<matrix 2> | Elementwise division of real or complex matrixes. |
| **.^** | <matrix 1>.^<matrix 2> | Elementwise raising of two matrixes to power or a matrix to real power. |
| **!** | <integer>! | Calculation of integer factorial |
| **..** | a..b | Calculation of an interval of integers from a to b.  **Example:**  1..4 means [1,2,3,4] |
| **#** | A**#**B | Reproduction of number B into a vector with dimension A.  **Example:**  3#0.1 means [0.1, 0.1, 0.1] |

Integral-valued and logical operators:

|  |  |  |
| --- | --- | --- |
| Operator | Semantics | Purpose and example |
| **or** | <expression 1> **or** <expression 2> | Bitwise logical operation OR. |
| **and** | <expression 1> **and** <expression 2> | Bitwise logical operation AND. |
| **xor** | <expression 1> **xor** <expression 2> | Bitwise logical operation EXCLUSIVE OR. |
| **div** | <dividend> **div** <divider> | Integer division. |
| **mod** | <dividend> **div** <divider> | Remainder of integer division. |
| **shl** | <number> **shl** <number of shifts> | Bitwise left shift. |
| **shr** | <number> **shr** <number of shifts> | Bitwise right shift. |

Special operators:

|  |  |  |
| --- | --- | --- |
| Operator | Semantics | Purpose and example |
| **()** | **(**<expression>**)** | Mathematical brackets. Bracketed expression is executed for the first time.  **Example:**  4\***(**10+7**)** |
| **()** | **(**<real part>,<virtual part>**)** | Packing of real numbers (vectors, matrixes) into a complex one (complex vector, matrix). Allowable writing of complex number parts with a space or semicolon.  **Example:**  **(**10, 5**)** |
| **[]** | **[**<element 1>{,<element 2>}**]** | Packing of elements into an array. If elements are real or integer numbers the result will be vector of real numbers. If elements are vectors of real numbers the result will be a matrix of real numbers. If elements are complex numbers the result will be vector of complex numbers. If elements are vectors of complex numbers the result will be a matrix of complex numbers. Comma, semicolon or space can be used as a divider for elements.  ***Note:*** If vector element is a negative number (-0.23) then this element in the vector shall be separated from the previous one with a comma.  **Example:**  X = [0, -0.23 2 3] – correct.  X = [0 -0.23 2 3] – wrong.  Vectors and matrixes can be assigned – in this case elements shall be variable.  **Example 1:**  *Vector*  **[**1, 4.5, 7**]** or **[**1 4.5 7**]**  **Example 2:**  *Matrix*  **[[**1,2**],[**6,7**]]** or **[[**1 2**];[**6 7**]]**  **Example 3:**  *Assignment of array elements*  **[**a,b**]** = **[**1, 2**]** |
| **[]** | <array or matrix>**[**<element or string number>{,<column number>}**]** | It returns the value of an element with specified number. Numeration is started with 1. Type of returned value corresponds to the type of array elements. If one index is specified for a matrix, then matrix row-vector is returned, if 2 – number is returned. This operator can be assigned. Comma, semicolon or space can be used as a divider for elements.  **Example 1:**  *Access*  A=C**[**1,1**]**  k=arr**[**i**]**  **Example 2:**  *Assignment*  C**[**1,1**]** = 10  k**[**3**]** = 4 |

***Note:*** Array brackets [] can be used for declaration of real arrays and matrixes in the same way as for “New block” (see description of **var** key word).