מבוא למכה שפת מטלאב

סממטר Киי תשע"ד

مواد:

נספח א: דפי עזר לבחינה
נספח ב: תחביר בסיסי

**if**

```matlab
if expression
    statements
elseif expression
    statements
else
    statements
end
```

**switch**

```matlab
switch (value)
    case {v1, v2, ...., vn}
        statements
    case v3
        statements
    otherwise
        statements
end
```

**for**

```matlab
for var = array
    statements
end
```

**while**

```matlab
while expression
    statements
end
```
Matlab

**all**

For vectors, \( \text{all}(V) \) returns logical 1 (TRUE) if none of the elements of the vector are zero. Otherwise it returns logical 0 (FALSE). For matrices, \( \text{ALL}(X) \) operates on the columns of \( X \), returning a row vector of logical 1’s and 0’s. For N-D arrays, \( \text{ALL}(X) \) operates on the first non-singleton dimension.

\( \text{all}(X,\text{DIM}) \) works down the dimension \( \text{DIM} \). For example, \( \text{ALL}(X,1) \) works down the first dimension (the rows) of \( X \).

**any**

\( B = \text{any}(A) \) tests whether any of the elements along various dimensions of an array is a nonzero number or is logical 1 (true). \( \text{any} \) ignores entries that are NaN (Not a Number).

If \( A \) is a vector, \( \text{any}(A) \) returns logical 1 (true) if any of the elements of \( A \) is a nonzero number or is logical 1 (true), and returns logical 0 (false) if all the elements are zero.

If \( A \) is a matrix, \( \text{any}(A) \) treats the columns of \( A \) as vectors, returning a row vector of logical 1’s and 0’s.

**ceil**

\( B = \text{ceil}(A) \) rounds the elements of \( A \) to the nearest integers greater than or equal to \( A \).

For complex \( A \), the imaginary and real parts are rounded independently.

**char**

\( S = \text{char}(A) \) converts the array \( A \) that contains nonnegative integers representing character codes into a MATLAB character array (the first codes are ASCII). The actual characters displayed depends on the 127 character encoding scheme for a given font. The result for any elements of \( A \) outside the range from 0 to 65535 is not defined (and may vary from platform to platform). Use DOUBLE to convert a character array into its numeric codes.

**find**

\( \text{ind} = \text{find}(X) \) locates all nonzero elements of array \( X \), and returns the linear indices of those elements in vector \( \text{ind} \). If \( X \) is a row vector, then \( \text{ind} \) is a row vector; otherwise, \( \text{ind} \) is a column vector. If \( X \) contains no nonzero elements or is an empty array, then \( \text{ind} \) is an empty array.

\( \text{ind} = \text{find}(X, k) \) or \( \text{ind} = \text{find}(X, k, \text{‘first’}) \) returns at most the first \( k \) indices corresponding to the nonzero entries of \( X \). \( k \) must be a positive integer, but it can be of any numeric data type.

\( \text{ind} = \text{find}(X, k, \text{‘last’}) \) returns at most the last \( k \) indices corresponding to the nonzero entries of \( X \).
[row, col] = find(X, ...) returns the row and column indices of the nonzero entries in the matrix X.
This syntax is especially useful when working with sparse matrices. If X is an N-dimensional array with N > 2, col contains linear indices for the columns.

floor

B = floor(A) rounds the elements of A to the nearest integers less than or equal to A. For complex A, the imaginary and real parts are rounded independently.

isempty

logical 0 (false) 1 (true) if A is an empty array and TF = isempty(A) returns logical one dimension of size zero, for example, 0-by- otherwise. An empty array has at least 0 or 0-by-5.

length

n = length(X) returns the size of the longest dimension of X. If X is a vector, this is the same as its length.

max

C = max(A) returns the largest elements along different dimensions of an array.
If A is a vector, max(A) returns the largest element in A.
If A is a matrix, max(A) treats the columns of A as vectors, returning a row vector containing the maximum element from each column.
If A is a multidimensional array, max(A) returns the maximum value of each vector.

min

C = min(A) returns the smallest elements along different dimensions of an array.
If A is a vector, min(A) returns the smallest element in A.
If A is a matrix, min(A) treats the columns of A as vectors, returning a row vector containing the minimum element from each column.
If A is a multidimensional array, min operates along the first nonsingleton dimension.
C = min(A,[],dim) returns the smallest elements along the dimension of A specified by scalar dim.
For example, min(A,[],1) produces the minimum values along the first dimension (the rows) of A.
**mod**

\[ M = \text{mod}(X,Y) \] if \( Y \neq 0 \), returns \( X - n*Y \) where \( n = \text{floor}(X/Y) \). If \( Y \) is not an integer and the quotient \( X/Y \) is within roundoff error of an integer, then \( n \) is that integer.

The inputs \( X \) and \( Y \) must be real arrays of the same size, or real scalars.

Example:

\[
X = [23 \\ 24 \\ 25 \\ 26] \\
M = \text{mod}(X, 3) \\
\text{Then } M \text{ gets the array } [2 \ 0 \ 1 \ 2]
\]

**ones**

\( Y = \text{ones}(n) \) returns an \( n \)-by-\( n \) matrix of 1s. An error message appears if \( n \) is not a scalar.

\( Y = \text{ones}(m,n) \) or \( Y = \text{ones}([m \ n]) \) returns an \( m \)-by-\( n \) matrix of ones.

\( Y = \text{ones}(m,n,p,\ldots) \) or \( Y = \text{ones}([m \ n \ p \ \ldots]) \) returns an \( m \)-by-\( n \)-by-\( p \)-by-\( \ldots \) array of 1s.

**repmat**

\( B = \text{repmat}(A,m,n) \) creates a large matrix \( B \) consisting of an \( m \)-by-\( n \) tiling of copies of \( A \). The size of \( B \) is \([\text{size}(A,1)*m, \text{size}(A,2)*n]\).

The statement \( \text{repmat}(A,n) \) creates an \( n \)-by-\( n \) tiling.

\( B = \text{repmat}(A,[m \ n \ p \ \ldots]) \) produces a multidimensional array \( B \) composed of copies of \( A \).

The size of \( B \) is \([\text{size}(A,1)*m, \text{size}(A,2)*n, \text{size}(A,3)*p, \ \ldots]\).

**reshape**

\( B = \text{reshape}(X,M,N) \) returns the \( M \)-by-\( N \) matrix whose elements are taken columnwise from \( X \). An error results if \( X \) does not have \( M*N \) elements.

**size**

\( d = \text{size}(X) \) returns the sizes of each dimension of array \( X \) in a vector \( d \) with \( \text{ndims}(X) \) elements. If \( X \) is a scalar, which MATLAB regards as a 1-by-1 array, \( \text{size}(X) \) returns the vector [1 1].

\([m,n] = \text{size}(X) \) returns the size of matrix \( X \) in separate variables \( m \) and \( n \).

\( m = \text{size}(X,\text{dim}) \) returns the size of the dimension of \( X \) specified by scalar \( \text{dim} \).

\([d1,d2,d3,\ldots,dn] = \text{size}(X) \), for \( n > 1 \), returns the sizes of the dimensions of the array \( X \) in the variables \( d1,d2,d3,\ldots,dn \), provided the number of output arguments \( n \) equals \( \text{ndims}(X) \).
**sum**

B = sum(A) returns sums along different dimensions of an array.
If A is a vector, sum(A) returns the sum of the elements.
If A is a matrix, sum(A) treats the columns of A as vectors, returning a row vector of the sums of each column.
If A is a multidimensional array, sum(A) treats the values along the first non-singleton dimension as vectors, returning an array of row vectors.
B = sum(A,dim) sums along the dimension of A specified by scalar dim. The dim input is an integer value from 1 to N, where N is the number of dimensions in A. Set dim to 1 to compute the sum of each column, 2 to sum rows, etc.

**zeros**

B = zeros(n) returns an n-by-n matrix of zeros. An error message appears if n is not a scalar.
B = zeros(m,n) or B = zeros([m n]) returns an m-by-n matrix of zeros.
B = zeros(m,n,p,...) or B = zeros([m n p ...]) returns an m-by-n-by-p-by-... array of zeros.