Class_Work_8

Contents

General

```
clc;clear all; format compact
```

Flow control

MATLAB has a logical data type, with the possible values 1, representing true, and 0, representing false.

```
a=true
b=1
c='1'
islogical(a) %To test whether a is logical
islogical(b)
```

```
a =
     1
b =
     1
c =
1
ans =
     1
ans =
     0
```

Convert numeric values to logical

returns an array that can be used for logical indexing or logical tests.

```
clc;clear all;
c=5;
c1=logical(c)
r3=[2 3 0 6 7]
r4=logical(r3) % r4 is a logical array that is the same size as r3
```

```
c1 =
     1
r3 =
     2     3     0     6     7
r4 =
     1     1     0     1     1
```

convert from logical to numeric

Most arithmetic operations remove the logicalness from an array.

```
%For example, adding zero to a logical array removes its logical characteristic. A = +A is
a=true
a1=+a
a2=a+0
```

```
a =
     1
a1 =
     1
a2 =
     1
```

Relational Operation

The relational operators are <, >, <=, >=, ==, and ~=. Whenever MATLAB encounters a relational operator, it produces a one if the expression is true and a zero if the expression is false

```
clc;clear all;
a1=4>2
a2=4<2
a3=5~=log(2)
a4=4>=4+1%arithmetic operation before Relational Operation
a5=(4>=4)+1
```

```
a1 =
     1
a2 =
     0
a3 =
     1
a4 =
     0
a5 =
     2
```

Relational operation between scalar and array

```
g1=[1 2 3]
f2=g1==2
B=magic(3)
f1=B<B(1,3)
```

```
g1 =
     1     2     3
f2 =
     0     1     0
B =
     8     1     6
     3     5     7
     4     9     2
f1 =
     0     1     0
     1     1     0
     1     0     1
```

Relational operators can be used on arrays,

as long as they are of the same size. Operations are performed element-by-element, resulting an array with ones in positions for which the relation was true and zeros in positions for which the relation was false.

```
a=[2:2:14]
b=[(1:7).^2]
c=a~=b
A=[0 1 0; 3 5 0; 0 0 2]; B=[8 0 6;0 5 0; 0 0 2];
g1=A>=B
g2=B<A
g3=A==B
a4='dot';b4='dog'% Relational operation between two strings
c=a4==b4
```

```
a =
     2     4     6     8    10    12    14
b =
     1     4     9    16    25    36    49
c =
     1     0     1     1     1     1     1
g1 =
     0     1     0
     1     1     1
     1     1     1
g2 =
     0     1     0
     1     0     0
     0     0     0
g3 =
     0     0     0
     0     1     1
     1     1     1
b4 =
dog
c =
     1     1     0
```

class_assignment_8,1

```
clc;clear all;
ex1=5<=8-31 %since 5<=5
ex2=7<3-1+6>2 % 0 since 7<8>2=(7<8)>2=1>2=0
ex3=(7<3)-1+(6>2) % 0 since 0-1+1=0. ex3 is numeric variable.
ex4=2*4+5==7+20/4 % 0 since 13~=12
```

```
ex1 =
     0
ex2 =
     0
```

```
ex3 =
     0
ex4 =
     0
```

class_assignment_8,2

```
clc; clear all
a=10; b=6;
ex1=a>=b  % 1 since 10>=6
ex2=a-b<=b/2  % 0 since 4>3
ex3=a-(b<=b/2)    % 10 since 10-0=10
```

```
ex1 =
     1
ex2 =
     0
ex3 =
    10
```

class_assignment_8,3

```
clc; clear all;
v=[4 -2 -1 5 0 1 -3 8 2];
w=[0 2 1 -1 0 -2 4 3 2];
v>=w
w~=v
```

```
ans =
     1     0     0     1     1     1     0     1     1
ans =
     1     1     1     1     0     1     1     1     0
```

class_assignment_8,4

```
y1=w(w>v)
y2=w(find(w>v)) % the same as y1
```

```
y1 =
     2     1     4
y2 =
     2     1     4
```

Find

Find indices and values of nonzero elements

```
clc;clear all;format compact
x=[1 0 4 -3 0 0 8 6]
indices=find(x)
y=x>2
x1=find(y)
%The result of find can then be used to extract just those elements of the vector
x2=x(x1)
% When find is applied to a matrix A, the index vector corresponds to A regarded
% as a vector of the columns stacked one on top of the other (that is, A(:)), and this
% vector can be used to index into A.
A=[1 0 4 ;-3 0 0 ;9 8 6]
g1=find(A)
g2=x(x>3)
```

```
g3=x(find(x>3));
```

```
x =
     1     0     4    -3     0     0     8     6
indices =
     1     3     4     7     8
y =
     0     0     1     0     0     0     1     1
x1 =
     3     7     8
x2 =
     4     8     6
A =
     1     0     4
    -3     0     0
     9     8     6
g1 =
     1
     2
     3
     6
     7
     9
g2 =
     4     8     6
```

logical operation & | ~

The symbols &, |, and ~ are the logical operators AND, OR, and NOT.

```
 %  The precedence for the logical operators with respect to each other is:
%  1. ~not has the highest precedence.
%  2.  & and.
% 3. or | has the lowest precedence.
clc;clear all;
a=1:3
b=3:-1:1
c=(b~=3)&(a>=b)
```

```
a =
     1     2     3
b =
     3     2     1
c =
     0     1     1
```

A value of zero means false, any non-zero value is considered true.

```
a=1;b=2;c=3;d=4;
z=(3<5)|(4==7)
z1=~3==7|4==6
```

```
z =
     1
z1 =
     0
```

class_assignment_8,4

```
clc;clear all;
ex1=5&-2    % 1 since 1&1=1 (all non zero are true)
ex2=8-2|6+5&~2  % 1 since 6|11&~2=1|1&0=1|0=1
ex3=~(4&0)+8*~(4|0) % 1 since ~0+8*~1=1+8*0=1
```

```
ex1 =
     1
ex2 =
     1
ex3 =
     1
```

class_assignment_8,6

```
clc;clear all
A=[1 4 8;9 3 6;7 4 2];
B=[3 1 9;5 2 8;2 1 7];
AgtB=(A>B).*A;  % AgtB => A bigger than B
```

class_assignment_8,7

```
clc;
A=round(rand(6,6)*10);
y=change(A);
```

class_assignment_8,8

```
clc;
B=rand(2,2)*5;
y=egool(B,2)
```

```
b =
   449   203
   364   469
y =
    4.4900    2.0300
    3.6400    4.6900
y =
    4.4900    2.0300
    3.6400    4.6900
```

class_assignment_8,9

```
c=5i+3
[r  zavit]=merucav(c)
```

```
c =
   3.0000 + 5.0000i
r =
    5.8310
zavit =
    1.0304
```

class_assignment_8,10

```
a=5;b=-3;c=19;
[y1 y2]=shoresh(a,b,c)
```

```
y1 =
   0.3000 + 1.9261i
y2 =
   0.3000 - 1.9261i
```

Published with MATLAB® 7.6