# Relational and Logical Operators

## Relational Operators

- Relational operators are used to represent conditions (such as “space $\leq$ 0” in the water tank example)
- Result of the condition is either true or false
- In MATLAB:
  - false is represented by 0
  - true is represented by 1 (non-zero)

# Relational Operators

| Operator | Description |
|---|---|
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |
| == | Equal to |
| ~= | Not equal to |

## Relational Operators

| **Operation** | **Result** |
|---|---|
| 3 < 4 | 1 |
| 3 <= 4 | 1 |
| 3 == 4 | 0 |
| 3 ~= 4 | 1 |
| 3 > 4 | 0 |
| 4 >= 4 | 1 |
| ‘A’ < ‘B’ | 1 |

# Relational Operators

- Don’t confuse equivalance (==) with assignment (=)
- Relational operations have lower priority than arithmetic operations (use parentheses to be safe, though)
- Relational operators are used as arithmetic operators within a mathematical expression
  - >>y=(6<10)+(7>8)+(5*3==60/4)
    - » 1        +      0     +              1
  - y=
  -     2

## Relational Operators

Suppose that x = [6,3,9] and y = [14,2,9]. The following MATLAB session shows some examples.

```
>>z = (x < y)
z =
   1    0    0
>>z = (x ~= y)
z =
   1    1    0
>>z = (x > 8)
z =
   0    0    1
```

# Relational Operators

The relational operators can be used for array addressing.

For example, with **`x = [6,3,9]`** and **`y = [14,2,9]`**, typing

```
z = x(x<y)
```

finds all the elements in x that are less than the corresponding elements in y. The result is z = 6.

# Relational Operators

- >>5>8
- ans=
-       0
- >>a=5<10
- a=
-    1
- >>b=[15 6 9 4 11 7 14]
- >>c=[8 20 9 2 19 7 10]
- >>d=c>=b

  Checks which c elements are larger or equal to b elements
- d=
-     0 1 1 0 1 1 0

- >>b==c

  Checks which b elements are equal to c elements
- ans=
-      0 0 1 0 0 1 0
- >>b~=c

  Checks which b elements are not equal to c elements
- ans=
-      1 1 0 1 1 0 1
- >>f= b-c>0
- f=
-  1 0 0 1 0 0 1
- Subtracts c from b and then checks which elements are larger than zero

## Relational Operators

- >>a=[2 9 4;-3 5 2;6 7 –1]
- A=
-  2 9 4
- -3 5 2
-  6 7 -1
- >>B=A<=2

  Checks which A elements are smaller or equal to 2. Assigns results to B
- B=
-  1 0 0
-  1 0 1
-  0 0 1

- >>3+4<16/2

  + and / are executed first
- ans=
-  1
- >>3+(4<16)/2
- ans=
-  ?

## Logical Operators

- Logical variables may have only the values 1 (true) and 0 (false).
- Logical operators:

| | |
|---|---|
| & | AND |
| \| | OR |
| xor | Exclusive OR |
| ~ | NOT |

## Logical Operators

| **input** | | **and** | **or** | **xor** | **not** |
|---|---|---|---|---|---|
| **a** | **b** | **a & b** | **a \| b** | **xor(a,b)** | **~a** |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

# Logical Operators

- >>3&7
- ans=
- 1

3 and 7 are both true (nonzero) so outcome is 1

- >>x=[9 3 0 11 0 15];
- >>y=[2 0 13 –11 0 4];
- >>~25
- ans=
- 0

25 is nonzero that is true and opposite is false (0)

- >>x&y
- ans=
- 1 0 0 1 0 1
- >>~(x+y)
- ans=
- 0 0 0 1 1 0
-

- >>z=x|y
- ans=
- 1 1 1 1 0 1

- >>t=25*((12&0)+(~0)+(0|5))

Use of logical operators in math expression

- t=
- 50

# Operator Hierarchy

- Processing order of operations:
  - parenthesis (starting from the innermost)
  - exponentials (left to right)
  - ~ operator
  - multiplications and divisions (left to right)
  - additions and subtractions (left to right)
  - relational operators (left to right)
  - & operator (left to right)
  - | operator (left to right)

## Accessing Arrays Using Logical Arrays

When a logical array is used to address another array, it extracts from that array the elements in the locations where the logical array has 1s.

So typing A(B), where B is a logical array of the same size as A, returns the values of A at the indices where B is 1.

## Accessing Arrays Using Logical Arrays

Just because an array contains only 0s and 1s, however, it is not necessarily a logical array. For example, in the following session k and w appear the same, but k is a logical array and w is a numeric array, and thus an error message is issued.

```
>>x = [-2:2]; k = (abs(x)>1)
k =
1   0   0    0    1
>>z = x(k)
z =
-2   2
>>w = [1,0,0,0,1]; v = x(w)
??? Subscript indices must either be real
positive integers or logicals.
```

# Built-in Logical Functions

- MATLAB built-in logical functions are equivalent to logical operators
  - and(A,B) equivalent to A&B
  - or(A,B) equivalent to A|B
  - not(A) equivalent to ~A
  - xor(A,B) exclusive OR,returns true if one operand is true and the other is false

# Built-in Logical Functions

- isempty(X) returns 1 if X is an empty array and 0 otherwise
- isnumeric() returns 1 if X is numeric and 0 otherwise
- ischar() returns 1 if X is char and 0 otherwise
- isinf() returns 1 if given value equals inf
- isnan() returns 1 if given value equals NaN

# Built-in Logical Functions

all(x) returns a **scalar**, which is 1 if all the elements in the vector are nonzero and 0 otherwise.

all(A) returns a **row vector** having the same number of columns as the matrix A and containing ones and zeros, depending on whether or not the corresponding column of A has all nonzero elements.

any(x) returns a **scalar**, which is 1 if any of the elements in the vector x is nonzero and 0 otherwise.

any(A) returns a **row vector** having the same number of columns as A and containing ones and zeros, depending on whether or not the corresponding column of the matrix A contains any nonzero elements.

# The find Function

| | |
|---|---|
| `find(A)` | Computes an array containing the indices of the nonzero elements of the array `A` in the form of linear indices. |
| `[u,v,w] = find(A)`<br>`x=[u v]` | Computes the vectors `u` and `v` containing the row and column indices of the nonzero elements of the array `A` and computes the array `w` containing the values of the nonzero elements. The array `w` may be omitted.Define x as concatenation of u and v to see row and column values together |

# Logical Operators and the find Function

Consider the session

```
>>x = [5, -3, 0, 0, 8];y = [2, 4, 0, 5, 7];
>>z = find(x&y)
z =
     1     2     5
```

**Note that the find function returns the indices, and not the values.**

## Logical Operators and the find Function

Remember, the find function returns the indices, and not the values. In the following session, note the difference between the result obtained by `y(x&y)` and the result obtained by `find(x&y)` in the previous slide.

```
>>x = [5, -3, 0, 0, 8];
>>y = [2, 4, 0, 5, 7];
>>values = y(x&y)
values =
2    4     7
>>how_many = length(values)
how_many =
3
```