

# 第二章 MATLAB基础

## ● 第三节 MATLAB数据类型及计算

### 一. 数据类型和全局变量

#### ❖ 数据类型:

Matlab是一种面向矩阵的编程语言，因此它任何数据都看做是矩阵，在Matlab中数据关系如下

- MATLAB的基本数据类型是双精度数据类型和字符类型
- MATLAB的不同数据类型的变量或对象占用的内存空间不同
- 不同的数据类型的变量或对象也具有不同的操作函数

# 1.基本数值类型

数据类型	说 明	字节数
double	双精度数据类型	8
sparse	稀疏矩阵数据类型	N/A
single	单精度数据类型	4
uint8	无符号8位整数	1
uint16	无符号16位整数	2
uint32	无符号32位整数	4
uint64	无符号64位整数	8
int8	有符号8位整数	1
int16	有符号16位整数	2
int32	有符号32位整数	4
int64	有符号64位整数	8

## 基本数值类型（续）

例 >> A=[1 2 3];

>> class(A)

ans =

double

>> whos

Name	Size	Bytes	Class
A	1x3	24	double array
ans	1x6	12	char array

Grand total is 9 elements using 36 bytes

>> B=int16(A);

>> class(B)

ans =

int16

>> whos

Name	Size	Bytes	Class
A	1x3	24	double array
B	1x3	6	int16 array
ans	1x5	10	char array

Grand total is 11 elements using 40 bytes

## 2. 整数类型数据运算

### 整数类型数据的运算函数

函 数	说 明
bitand	数据位“与”运算
bitcmp	按照指定的数据位数求数据的补码
bitor	数据位“或”运算
bitmax	最大的浮点整数数值
bitxor	数据位“异或”运算
bitset	将指定的数据位设置为1
bitget	获取指定的数据位数值
bitshift	数据位移操作

注意：参与整数运算的数据都必须大于0

## 整数类型数据运算（续）

例：数据位操作（bitset函数）

```
>> A=86;
>> dec2bin(A)
ans =
1010110
>> B=bitset(A,6)
B =
    118
>> dec2bin(B)
ans =
1110110
>> C=bitset(A,7,0)
C =
    22
>> dec2bin(C)
ans =
10110
```

**bitset(A,B,C)**  
函数根据输入的第二个参数设置相应的数据位的数值，若不指定第三个参数，则将相应的数据位设置为“1”，否则根据输入的第三个参数设置相应的数据位。

```
>> A=86
A =
    86
>> B=bitset(A,6)
B =
    118
>> C=bitset(A,7,0)
C =
    22
```

## ❖ 全局变量：用global定义

global X Y Z, 则X、Y、Z就会被定义为全局变量。

- ❖ 函数文件的内部变量是局部的，与其它的函数文件及MATLAB内存相互隔离。
- ❖ 如果在某些函数中，都把某一变量定义为全局变量，那么这些函数将公用这个变量。
- ❖ 全局变量的作用域是整个MATLAB的工作区，即全程有效，所有的函数都可以对它们进行存取和修改。因此，定义全局变量是函数间传递数据的一个手段。
- ❖ 全局变量将给程序调试和维护带来不便。

# 第二章 MATLAB基础

## ● 第三节 MATLAB数据类型及计算

### 二. 逻辑类型和关系运算

- 逻辑数据类型
- 逻辑运算
- 关系运算
- 运算符的优先级



# 1 逻辑数据类型

- 逻辑数据类型
  - 逻辑数据类型就是仅具有两个数值的一种数据类型
    - True —— 用1表示
    - False —— 用0表示
- 任何数值都可以参与逻辑运算
  - 非零值看作逻辑真
  - 零值看作逻辑假
- 逻辑类型的数据只能通过数值类型转换，或者使用特殊的函数生成相应类型的数组或者矩阵

## 逻辑数据类型（续）

```
>> D=false([size(A),2])
```

?

例: >> A=eye(3)

A =  
1 0 0  
0 1 0  
0 0 1

```
>> B=logical(A)
```

B =  
1 0 0  
0 1 0  
0 0 1

```
>> C=true(size(A))
```

C =  
1 1 1  
1 1 1  
1 1 1

```
>> C=true(3,3)
```

## 逻辑数据类型（续）

例: >> A=eye(3)

A =  
1 0 0  
0 1 0  
0 0 1

>> B=logical(A)

B =  
1 0 0  
0 1 0  
0 0 1

>> C=true(size(A))

C =  
1 1 1  
1 1 1  
1 1 1

>> D=false([size(A),2])

D(:,:,1) =  
0 0 0  
0 0 0  
0 0 0

D(:,:,2) =  
0 0 0  
0 0 0  
0 0 0

>> C=true(3,3)

## 逻辑数据类型（续）

例: >> A=eye(3)

A =  
1 0 0  
0 1 0  
0 0 1

>> B=logical(A)

B =  
1 0 0  
0 1 0  
0 0 1

>> C=true(size(A))

C =  
1 1 1  
1 1 1  
1 1 1

>> C=true(3,3)

>> D=false([size(A),2])

D(:,:,1) =

0 0 0  
0 0 0  
0 0 0

D(:,:,2) =

0 0 0  
0 0 0  
0 0 0

>> whos

Name	Size
A	3x3
B	3x3
C	3x3
D	3x3x2

逻辑类型的数组  
每一个元素  
仅占用一个字  
节的内存空间

Bytes	Class
72	double array
9	logical array
9	logical array
18	logical array

Grand total is 45 elements using 108 bytes

## 逻辑数据类型（续）

- 在使用true或者false函数创建逻辑类型数组时，若不指明参数，则创建一个逻辑类型的标量

例 >> a=true

a =

1

>> b=false

b =

0

>> c=1

c =

1

>> isnumeric(a)

ans =

0

>> isnumeric(c)

ans =

1

>> islogical(a)

ans =

1

>> islogical(b)

ans =

1

>> islogical(c)

ans =

0

在MATLAB中有些函数以is 开头，这类函数是用来完成某种判断功能的函数。

isnumeric(\*):判断输入  
的参数是否为数值类型  
islogical(\*):判断输入的  
参数是否为逻辑类型

## 2 逻辑运算

- 能够处理逻辑类型数据的运算叫作逻辑运算

### MATLAB的逻辑运算

运算符	说明
& &	具有短路作用的逻辑与操作，仅能处理标量
	具有短路作用的逻辑或操作，仅能处理标量
&	元素与操作
	元素或操作
~	逻辑非操作
xor	逻辑异或操作
any	当向量中的元素有非零元素时，返回真
all	当向量中的元素都是非零元素时，返回真

## 逻辑运算（续）

- 参与逻辑运算的操作数不一定必须是逻辑类型的变量或常量，其他类型的数据也可以进行逻辑运算，但运算结果一定是逻辑类型的数据

例：>> a=eye(3)

a =

1	0	0
0	1	0
0	0	1

>> b=a;b(3,1)=1

b =

1	0	0
0	1	0
1	0	1

>> a&b

ans =

1	0	0
0	1	0
0	0	1

>> whos

Name	Size	Bytes	Class
a	3x3	72	double array
ans	3x3	9	logical array
b	3x3	72	double array
Grand total is 27 elements using 153 bytes			

## 逻辑运算（续）

- 具有短路作用的逻辑“与”操作（&&）和“或”操作（||）
  - 进行 `a && b && c && d` 运算时，若 `a` 为假（0），则后面的三个变量都不再被处理，运算结束，并返回运算结果逻辑假（0）
  - 进行 `a || b || c || d` 运算时，若 `a` 为真（1），则后面的三个变量都不再被处理，运算结束，并返回运算结果逻辑真（1）
  - 仅能处理标量



## 逻辑运算（续）

例: >> a=eye(3)

a =

1	0	0
0	1	0
0	0	1

>> b=a;b(3,1)=1

b =

1	0	0
0	1	0
1	0	1

>> a&& b

??? Operands to the || and && operators must be convertible to logical scalar values.

## 逻辑运算（续）

例： >> a=0;b=1;c=2;d=3;

>> a&&b&&c&&d

ans =

0

>> a=0;b=2;c=6;d=8;

>> a&&b&&c&&d

ans =

0

>> a=10;b=1;c=2;d=3;

>> a||b||c||d

ans =

1

>> a=10;b=0;c=7;d=9;

>> a||b||c||d

ans =

1

>> whos

Name	Size	Bytes	Class
a	1x1	8	double array
ans	1x1	1	logical array
b	1x1	8	double array
c	1x1	8	double array
d	1x1	8	double array

Grand total is 5 elements using 33 bytes

## 逻辑运算（续）

例：函数any和all的使用示例（对向量）

```
>> a=[1 2 3 0];
```

```
>> any(a)
```

```
ans =
```

```
1
```

```
>> all(a)
```

```
ans =
```

```
0
```

```
>> b=[0 0 0 0];
```

```
>> any(b)
```

```
ans =
```

```
0
```

```
>> all(b)
```

```
ans =
```

```
0
```

```
>> c=[1 2 3 4];
```

```
>> any(c)
```

```
ans =
```

```
1
```

```
>> all(c)
```

```
ans =
```

```
1
```

### 3 关系运算

- 关系运算是用来判断两个操作数关系的运算  
MATLAB的关系运算符

运算符	说明
==	等于
~=	不等于
<	小于
>	大于
<=	小于等于
>=	大于等于

## 关系运算（续）

- 参与关系运算的操作数可以是各种数据类型的变量或者常数
- 运算结果是逻辑类型的数据
- 标量可以和数组（或矩阵）进行比较，比较时自动扩展标量，返回的结果是和数组同维的逻辑类型数组
- 若比较的是两个数组，则数组必须是同维的，且每一维的尺寸必须一致

## 关系运算（续）

例: >> A=reshape(1:9,3,3)

A =

1	4	7
2	5	8
3	6	9

>> B=magic(3)

B =

8	1	6
3	5	7
4	9	2

>> A>B

ans =

0	1	1
0	0	1
0	0	1

>> whos

Name	Size	Bytes	Class
A	3x3	72	double array
B	3x3	72	double array
ans	3x3	9	logical array

Grand total is 27 elements using  
153 bytes

## 关系运算（续）

例: >> a=4

a =  
4

>> A=reshape(1:9,3,3)

A =

1	4	7
2	5	8
3	6	9

>> a>A

ans =

1	0	0
1	0	0
1	0	0



4	4	4
4	4	4
4	4	4

>> whos

Name	Size	Bytes	Class
A	3x3	72	double array
a	1x1	8	double array
ans	3x3	9	logical array

Grand total is 19 elements  
using 89 bytes

## 关系运算（续）

- 利用“（）”和各种运算符相结合，可以完成复杂的关系运算

例：>> A=reshape(-4:4,3,3)

A =

```
-4  -1   2
-3   0   3
-2   1   4
```

>> A>=0

ans =

```
0   0   1
0   1   1
0   1   1
```

>> B=~(A>=0)

B =

```
1   1   0
1   0   0
1   0   0
```

>> whos

Name	Size	Bytes	Class
A	3x3	72	double array
B	3x3	9	logical array
ans	3x3	9	logical array

Grand total is 27 elements using 90 bytes



## 关系运算（续）

例：>> C=(A>0)&(A<3)

C =

0	0	1
0	0	0
0	1	0

过程：>> A>0

ans =

0	0	1
0	0	1
0	1	1

>> A<3

ans =


1	1	1
1	1	0
1	1	0

A =

-4	-1	2
-3	0	3
-2	1	4

## 关系运算（续）

- 逻辑索引：将逻辑类型的数据应用于索引就构成了逻辑索引
  - 利用逻辑索引可以方便地从矩阵或者数组中找到某些符合条件的元素

例：>> A=[-2 10 NaN 30 0 -11 -Inf 31];  
>> pos=A<0  
pos =  
1 0 0 0 0 1 1 0  
>> B=A(pos)  >> B=A(A<0)  
B =  
-2 -11 -Inf  
>> pos=(A>=0)&(isfinite(A))  
pos =  
0 1 0 1 1 0 0 1  
>> C=A(pos)  
C =  
10 30 0 31

## 4 运算符的优先级

- (1) 括号 ( )
- (2) 数组转置 (.'), 数组幂 (.^), 矩阵转置 ('), 矩阵幂 (^)
- (3) 一元加 (+), 一元减 (-), 逻辑非 (~)
- (4) 数组乘法 (.\*), 数组右除 (./), 数组左除 (.\)  
矩阵乘法 (\*), 矩阵右除 (/), 矩阵左除 (\)
- (5) 加法 (+), 减法 (-)
- (6) 冒号运算符 ( : )
- (7) 小于 (<), 小于等于 (<=), 大于 (>), 大于等于 (>=)  
等于 (==), 不等于 (~=)
- (8) 元素与 (&)
- (9) 元素或 (|)
- (10) 短路逻辑与 (&&)
- (11) 短路逻辑或 (||)

# 三.字符与字符串

- 字符串在数据的可视化、应用程序的交互方面起到非常重要的作用
- 创建字符串时需要使用单引号将字符串的内容包括起来
- 字符串一般以行向量形式存在，并且每一个字符占用两个字节的内存
- 主要内容
  - 1. 字符串规则
  - 2. 基本字符串操作
  - 3. 字符串操作函数
  - 4. 字符串转换函数
  - 5. 格式化输入输出

### 三.字符与字符串

#### ● 1.字符串规则

- ❖ 在Matlab中所有字符串都用单引号界定后输入或赋值。如`s='hello'`。
- ❖ 字符串的每个字符（空格也是字符）都是响应矩阵的一个元素。如`s`是 $1 \times 5$ 的矩阵，可用`size(s)`查得。
- ❖ 字符以`ascii`码储存。用`abs`指令可以看到字符的`ascii`码值。  
如`abs(s)`可以得到如下结果  
`ans=72 101 108 108 111。`

## ● 1. 字符串规则

- ❖ 可以用指令`setstr`实现ASCII码值向字符的转换。
- ❖ 字符变量可以用方括号合并合并成更大的“串”。例如运行`s=[s, 'world']`,  
可得`s=Hello world`
- ❖ 用`eval/feval`函数将字符变量转换为宏功能。  
`eval(t)/feval(t)`就是运行包含在`t`中的内容。
- `Eval(expression)`执行字符串`expression`所表达的式子，用户可以用`[]`连接字符串和变量来构造`expression`。
- `Feval`则不同，它以输入量作为某个函数的函数名来执行。

# ● 1.字符串规则

## ■ 创建字符串的方法

- 创建字符串时，只要将字符串的内容用单引号包括起来即可

例: >> a=127

a =

127

>> class(a)

ans =

double

>> size(a)

ans =

1 1

>> b='127'

b =

127

>> class(b)

ans =

char

>> size(b)

ans =

1 3

# ● 1.字符串规则

- 若需要在字符串内容中包含单引号，则在键入字符串内容时，连续键入两个单引号即可

例： >> C='Isn't it?'

C =

Isn't it?



# ● 1.字符串规则

- 使用char函数创建一些无法通过键盘输入的字符
  - 该函数的作用是将输入的整数参数转变为相应的字符

```
>> S1=char('This string array','has two rows.')
```

```
S1 =
```

```
This string array
```

```
has two rows.
```

```
>> S2=char('这','字符','串数组','由4行组成')
```

```
S2 =
```

```
这
```

```
字符
```

```
串数组
```

```
由4行组成
```

## ● 2. 基本字符串操作

- 字符串元素索引
- 字符串的拼接
- 字符串与数值之间的转换

# (1) 字符串元素索引

- 字符串实际上也是一种MATLAB的向量或者数组，一般利用索引操作数组的方法都可以用来操作字符串

例：>> a='This is No.3-15 Example!'

a =

This is No.3-15 Example!

>> b=a(1:4)

b =

This

>> c=a(12:15)

c =

3-15

>> d=a(17:end)

d =

Example!

> a='Hello MOTO!'

a =

Hello MOTO!

>> b=a(end:-1:1)

b =

!OTOM olleH

## (2) 字符串的拼接

- 字符串可以利用“[]”运算符进行拼接
  - 若使用“,”作为不同字符串之间的间隔,则相当于扩展字符串成为更长的字符串向量
  - 若使用“;”作为不同字符串之间的间隔,则相当于扩展字符串成为二维或者多维的数组,这时不同行上的字符串必须具有同样的长度

例: >> a='Hello';  
>> b='MOTO!';  
>> length(a)==length(b)  
ans =  
1  
>> c=[a,' ',b]  
c =  
Hello MOTO!  
>> d=[a;b]  
d =  
Hello

```
>> size(c)
ans =
     1    11
>> size(d)
ans =
     2     5
>> e='MOTO';
>> f=[a;e]
??? Error using ==> vertcat
All rows in the bracketed
expression must have the same
number of columns.
```

### (3) 字符串和数值的转换

- 使用char函数可以将数值转变为字符
- 使用double函数可以将字符转变成数值

例: >> a='Hello MOTO!';

>> b=double(a)

b =

72 101 108 108 111 32 77 79 84 79 33

>> c='您好!';

>> d=double(c)

d =

50426 47811 41889

>> char(b)

ans =

Hello MOTO!

>> char(d)

ans =

您好!

### ● 3. 字符串操作函数

函 数	说 明
char	创建字符串，将数值转变成为字符串
double	将字符串转变成为Unicode数值
ischar	判断变量是否是字符类型
strcat	水平组合字符串，构成更长的字符向量
strvcat	垂直组合字符串，构成字符串矩阵
strcmp	比较字符串，判断字符串是否一致
strncmp	比较字符串前n个字符，判断是否一致
strcmpi	比较字符串，比较时忽略字符的大小写
strncmpi	比较字符串前n个字符，比较时忽略字符的大小写
findstr	在较长的字符串中查寻较短的字符串出现的索引
strfind	在第一个字符串中查寻第二个字符串出现的索引
strjust	对齐排列字符串
strmatch	查询匹配的字符串

### ● 3.字符串操作函数

函数名	作用
isstr	判断是否为字符
blanks	空白字符
deblank	移去空白字符
eval	运行字符串
feval	运行字符串
strcmp	比较字符串
upper	将字符串变为大写形式
lower	将字符串变为小写形式
abs	将字符串变为ASCII码值
setstr	将ASCII码值变为字符串
num2str	将数字变为字符串
str2num	将字符串变为数字

### ● 3.字符串操作函数

函数名	作用
strrep	用一个字符串代替另一个字符串
findstr	从一个字符串中寻找是否包含另一个字符串
str2mat	将字符串变为文本矩阵
sprintf	将带格式的数字变为字符串
sscanf	将字符串转变为带格式的数字
hex2num	将十六进制的字符串转变为IEEE浮点数
hex2dec	将十六进制的字符串转变为十进制数
dec2hex	将十进制数转变为十六进制的字符串

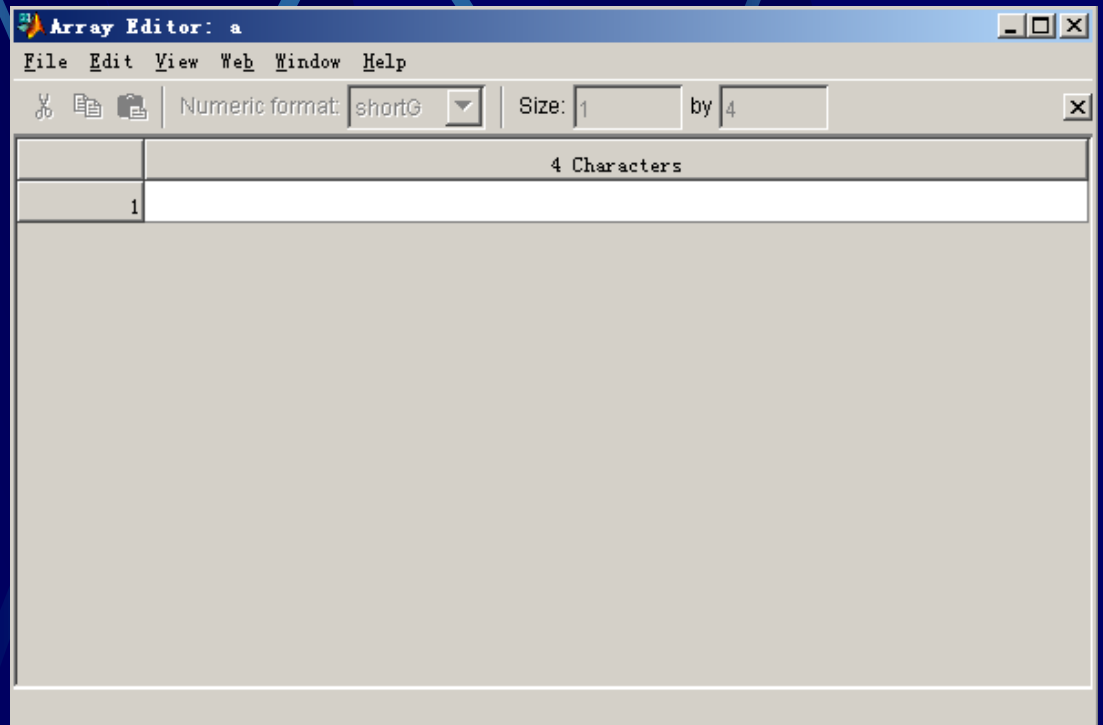


### ● 3. 字符串操作函数

例 (blanks) :

```
>> a=blanks(4)
```

```
a =
```



### ● 3. 字符串操作函数

例 (deblank) :

```
>> a='Hello! '
```

```
a =
```

```
Hello!
```

```
>> deblank(a)
```

```
ans =
```

```
Hello!
```

```
>> whos
```

Name	Size	Bytes	Class
a	1x9	18	char array
ans	1x6	12	char array

Grand total is 15 elements using 30 bytes

### ● 3. 字符串操作函数

例 (ischar) :

```
>> a='Hello!'
```

```
a =
```

```
Hello!
```

```
>> ischar(a)
```

```
ans =
```

```
1
```

```
>> b=12;
```

```
>> ischar(b)
```

```
ans =
```

```
0
```

变量为字符型，则结果为1

变量不为字符型，则结果为0

### ● 3. 字符串操作函数

例：组合字符串（**strcat**和**strvcat**）

```
>> a='Hello';  
>> b='MOTO!';  
>> c=strcat(a,b)  
c =  
HelloMOTO!  
>> d=strvcat(a,b ,c)  
d =  
Hello  
MOTO!  
HelloMOTO!
```

```
>> whos
```

Name	Size	Bytes	Class
a	1x5	10	char array
b	1x5	10	char array
c	1x10	20	char array
d	3x10	60	char array

Grand total is 50 elements using  
100 bytes

**Strvcat**函数允许将不同长度的字符串组合成为字符矩阵，并且将短字符串扩充为与长字符串相同的长度

### ● 3. 字符串操作函数

例：比较字符串（ strcmp和strncmp ）

```
>> a='The first string';
```

```
>> b='The second string';
```

```
>> c=strcmp(a,b)
```

```
c =
```

```
0
```

```
>> d=strncmp(a,b,4)
```

```
d =
```

### ● 3. 字符串操作函数

例：查寻索引（ findstr和strfind ）

```
>> S1='A friend in need is a friend indeed';
```

```
>> S2='friend';
```

```
>> a=findstr(S2,S1)
```

```
a =
```

```
3 23
```

```
>> b=strfind(S2,S1)
```

```
b =
```

```
[]
```

```
>> c=strfind(S1,S2)
```

```
c =
```

```
3 23
```

### ● 3. 字符串操作函数

例：对齐排列字符串（`findstr`和`strfind`）

```
>> a='Hello';
```

```
>> b='MOTO!';
```

```
>> c=strcat(a,b)
```

```
c =
```

```
HelloMOTO!
```

```
>> d=strvcat(a,b,c)
```

```
d =
```

```
Hello
```

```
MOTO!
```

```
HelloMOTO!
```

```
>> e=strjust(d)
```

```
e =
```

```
Hello
```

```
MOTO!
```

```
HelloMOTO!
```

### ● 3. 字符串操作函数

例：替换字符串中的子字符（ `strrep` ）

```
>> S1='A firend in need is a firend indeed'
```

```
S1 =
```

```
A firend in need is a firend indeed
```

```
>> S2=strrep(S1,'firend','friend')
```

```
S2 =
```

```
A friend in need is a friend indeed
```



### ● 3. 字符串操作函数

例：查寻匹配的字符串（`strmatch`）

```
>> a=strmatch('max',strvcat('max','minimax','maximum'))
```

```
a =
```

```
1
```

```
3
```

```
>> b=strmatch('max',strvcat('max','minimax','maximum'),'exact')
```

```
b =
```

```
1
```

### ● 3. 字符串操作函数

例：改变字符串的字符的大小写（ upper和lower ）

```
>> S1='friend';
```

```
>> S2=upper(S1)
```

```
S2 =
```

```
FRIEND
```

```
>> S3=lower(S2)
```

```
S3 =
```

```
friend
```

## ● 4. 字符串转换函数

- 在MATLAB中使用不同的函数可以允许不同类型的数据和字符串类型的数据之间进行转换
- 在MATLAB中直接提供了相应的函数对同样类型的数据进行数制的转换

## ● 4. 字符串转换函数

### 数字和字符之间的转换函数

函数	说明
num2str	将数字转变成为字符串
int2str	将整数转变成为字符串
mat2str	将矩阵转变成为可被eval函数使用的字符串
str2double	将字符串转变为双精度类型的数据
str2num	将字符串转变为数字
sprintf	格式化输出数据到命令行窗口
sscanf	读取格式化字符串

## ● 4. 字符串转换函数

### 不同数值之间的转换函数

函数	说明
hex2num	将十六进制整数字符串转变成为双精度数据
hex2dec	将十六进制整数字符串转变成为十进制整数
dec2hex	将十进制整数转变成为十六进制整数字符串
bin2dec	将二进制整数字符串转变成为十进制整数
dec2bin	将十进制整数转变成为二进制整数字符串
base2dec	将指定数制类型的数字字符串转变成为十进制整数
dec2base	将十进制整数转变成为指定数制类型的数字字符串

## 4. 字符串转换函数

函数`str2num`在使用时需要注意：

- 被转换的字符串仅能包含数字、小数点、字符“e”或者“d”、数字的正号或者负号、复数的虚部字符“i”或者“j”
- 使用时要注意空格

```
例： >> A=str2num('1+2i')
      A =
      1.0000 + 2.0000i
>> B=str2num('1 +2i')
      B =
      1.0000      0 + 2.0000i
>> C=str2num('1 + 2i')
      C =
      1.0000 + 2.0000i
```

```
>> whos
```

Name	Size	Bytes	Class
A	1x1	16	double array (complex)
B	1x2	32	double array (complex)
C	1x1	16	double array (complex)

Grand total is 4 elements using 64 bytes

可以使用`str2double`函数避免上述问题，但`str2double`函数只能转换标量，不能转换矩阵或者数组

## ● 4. 字符串转换函数

例:

```
>> S=['1 2';'2 3']
```

```
S =
```

```
1 2
```

```
2 3
```

```
>> A=str2num(S)
```

```
A =
```

```
1 2
```

```
2 3
```

```
>> whos
```

Name	Size	Bytes	Class
A	2x2	32	double array
S	2x3	12	char array

Grand total is 10 elements using 44 bytes

## ● 4. 字符串转换函数

- 使用函数num2str将数字转换为字符串时，可以指定字符串所表示的有效数字位数

例>> A=num2str(rand(2,2),4)

A =

0.8913    0.4565

0.7621    0.0185

>> B=num2str(rand(2,2),6)

B =

0.921813    0.176266

0.738207    0.405706



## 4. 字符串转换函数

### ■ 其他的转换函数示例

```
>> a=255;  
>> h=dec2hex(a)  
h =  
FF  
>> b=dec2bin(a)  
b =  
11111111  
>> c=dec2base(a,5)  
c =  
2010  
>> b(end)='0'  
b =  
11111110
```

```
>> bin2dec(b)
```

```
ans =  
254
```

```
>> whos
```

Name	Size	Bytes	Class
a	1x1	8	double array
ans	1x1	8	double array
b	1x8	16	char array
c	1x4	8	char array
h	1x2	4	char array

Grand total is 16 elements using 44 bytes