# 1124_课堂内容

## 数组的使用方法

数组的使用:

**添加元素几种常用方法**

```
>> A(4) = 4
A =
     1     2     3     4
```

✓ 直接使用下标进行元素添加（注：最好提前声明空间）

✓ 元素访问用的是()，而非我们常见的[]

思考：第一个元素如何访问

```
>> A(0)
数组索引必须为正整数或逻辑值。
```

```
>> A(1)
ans =
     1
```

数组的元素下标开始于1，而非我们常用的0!

```
>> A(end)
ans =
     4
```

```
>> A(end+1) = 5
A =
     1     2     3     4     5
```

这种方式机制相同，借助end关键字

```
% 添加数组元素的一些方法
A = [1,2,3,4,5,6];
A = [A,7] % 利用重构的方法
```

```
A = 1×7
     1     2     3     4     5     6     7
```

```
A = [A,[9,10]]
```

```
A = 1×9
     1     2     3     4     5     6     7     9    10
```

```
% 利用下标，在中间插入元素
B = [1,2,3,4,5,6];
B = [B(1:2),100,B(3:6)]
```

```
B = 1×7
     1     2   100     3     4     5     6
```

注意效率问题！ 思考：哪种添加元素方式的效率最高?

```
>> s = 500;
tic
A = [];
for i = 1:s
    A = [A,i];
end
toc
历时 0.005622 秒。
```

涉及移动元素，效率不高

```
>> s = 500;
tic
A = [];
for i = 1:s
    A(i) = i;
end
toc
历时 0.004217 秒。
```

下标赋值，效率可提升

```
>> s = 500;
tic
A = zeros(1,s);
for i = 1:s
    A(i) = i;
end
toc
历时 0.003572 秒。
```

预先声明空间，效率最好（s越大越明显）

```
% 访问
A = [1,2,3,4];
A(1)
```

```
ans = 1
```

```
A(1:3)
```

```
ans = 1×3
     1     2     3
```

```
A(end:-1:1) % 逆序输出 step = -1
```

```
ans = 1×4
     4     3     2     1
```

```
A(end:-2:1) % step = -2
```

```
ans = 1×2
     4     2
```

```
A(2:end) % 访问到最后的元素
```

```
ans = 1×3
     2     3     4
```

```
A([1,3]) % 同时访问多个元素
```

```
ans = 1×2
     1     3
```

```
% 赋值
A = [1,2,3,4]
```

```
A = 1×4
     1     2     3     4
```

```
A(2) = 0
```

```
A = 1×4
     1     0     3     4
```

```
A([1,2])=[0,0]
```

```
A = 1×4
     0     0     3     4
```

```
A([1:3])=5
```

```
A = 1×4
     5     5     5     4
```

```
A([1:2])=[6,7]
```

```
A = 1×4
     6     7     5     4
```

**特殊数组的创建**

- 通过冒号创建一维数组

调用格式为X = $N_1$:step:$N_2$，从第一个元素$N_1$开始，每次递增(step > 0)或递减step(step < 0)，直到最后一个元素与$N_2$差的绝对值小于或等于|step|为止，默认step为1

```
% 通过冒号创建一维数组
% 注意 -- 中间是 step
A = 1:5
```

```
A = 1×5
     1     2     3     4     5
```

```
B = 3.2:-0.1:0.1
```

```
B = 1×32
    3.2000    3.1000    3.0000    2.9000    2.8000    2.7000    2.6000    2.5000 · · ·
```

```
C = 3:2:1
```

```
C =

  空的 1×0 double 行向量
```

数组的使用:

**特殊数组的创建**

- logspace建立一维数组

**logspace** - 生成对数间距向量

此 MATLAB 函数 生成一个由在 10^a 和 10^b（10 的 N 次幂）之间的 50 个对数间距点组成的行向量 y。logspace 函数对于创建频率向量特别有用。该函数是 linspace 和:运算符的对数等价函数。

logspace(a, b)：在[$10^a$, $10^b$]区间内生成50个差值相等的数

longspace(a, b, n)：在[$10^a$, $10^b$]区间内生成n个差值相等的数

longspace(a, pi)：在[$10^a$, pi]区间内生成50个差值相等的数

```
>> logspace(0, 1, 5)

ans =

    1.0000    1.7783    3.1623    5.6234   10.0000
```

```
>> logspace(0, 2, 3)

ans =

     1    10   100
```

注意这块是闭区间!

```
logspace(0,1,5)
```

```
ans = 1×5
    1.0000    1.7783    3.1623    5.6234   10.0000
```

```
logspace(0,2,3)
```

```
ans = 1×3
     1    10   100
```

**特殊数组的创建**

- linspace建立一维数组

```
>> help linspace
linspace - 生成线性间距向量
   此 MATLAB 函数 返回包含 x1 和 x2 之间的 100 个等间距点的行向量。
```

linspace(a, b)：第一个元素为a，最后一个元素为b，总数为100的等差数列

linspace(a, b, n)：第一个元素为a，最后一个元素为b，有n个元素的等差数列

```
>> linspace(1, 10, 10)

ans =

     1     2     3     4     5     6     7     8     9    10
```

```
>> linspace(1, 2, 1)

ans =

     2
```

n < 2时，返回元素b

```
>> linspace(1, 2, 1)
```

思考：该式的输出是什么

```
>> linspace(1, 5, 0)

ans =

  空的 1×0 double 行向量
```

12

```
linspace(1,10,100)
```

```
ans = 1×100
    1.0000    1.0909    1.1818    1.2727    1.3636    1.4545    1.5455    1.6364 · · ·
```

```
linspace(1,2,1)
```

```
ans = 2
```

```
linspace(1,2,0)
```

```
ans =

  空的 1×0 double 行向量
```

## 常见运算

```
A = [1,2,3,4];
B = [5,6,7,8];
A + B
```

```
ans = 1×4
     6     8    10    12
```

```
A - B
```

```
ans = 1×4
    -4    -4    -4    -4
```

```
A + 1
```

```
ans = 1×4
     2     3     4     5
```

```
% C =[9,10]
% A + C -- 大小不兼容
```

## 数组的运算:

### 算术运算

加、减、乘、左除、右除和乘方

通过格式".*"或者"./"可实现数组乘除，要求数组维度相同

如：A./ B= B .\ A，其中A是被除数，B是除数

```
>> A = [1 2 3 4];
>> B = [5 6 7 8];
```

```
>> A .* B

ans =

     5    12    21    32
```

```
>> A ./ B

ans =

    0.2000    0.3333    0.4286    0.5000

>> B .\ A

ans =

    0.2000    0.3333    0.4286    0.5000
```

```
>> A./2

ans =

    0.5000    1.0000    1.5000    2.0000
```

```
>> A/2

ans =

    0.5000    1.0000    1.5000    2.0000
```

✓ 允许对于单一数操作

```
A = [1,2,3,4];
B = [5,6,7,8];
A.*B
```

```
ans = 1×4
     5    12    21    32
```

```
A./B
```

```
ans = 1×4
    0.2000    0.3333    0.4286    0.5000
```

```
B.\A
```

```
ans = 1×4
    0.2000    0.3333    0.4286    0.5000
```

### 算术运算

加、减、乘、左除、右除和乘方

通过乘方格式".^"实现数组乘方运算，包括数组间、数组与某个值的乘方预算

```
>> A = [1 2 3];
>> B = [3 4 5];
>> A.^B

ans =

     1    16   243
```

```
>> 2.^A

ans =

     2     4     8
```

```
>> A.^2

ans =

     1     4     9
```

✓ 注意操作符中的"."

```
>> A^B
错误使用 ^ (第 51 行)
用于对矩阵求幂的维度不正确。请检查并确保矩阵为方阵并且幂为标量。要执行按元素矩阵求幂，请使用 '.^'。
```

16

```
A = [1,2,3,4];
B = [5,6,7,8];
A.^B
```

```
ans = 1×4
```

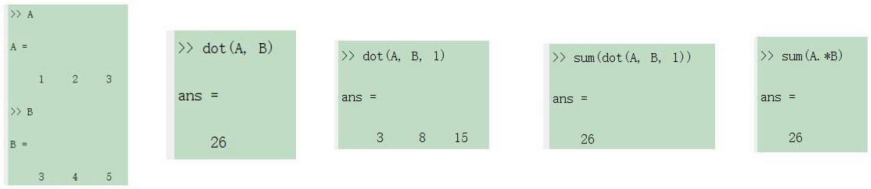
1 64 2187 65536

**算术运算**

加、减、乘、左除、右除和乘方

一般与dot函数等价，运算规则要求数组A和数组B维数相同

dot(A, B)：返回内积结果　　dot(A, B, dim)：返回内积结果，dim指定计算的维度




17

```
A = [1,2,3,4];
B = [5,6,7,8];
% dot(A,B) 返回内积结果
% dot(A,B,dim) 返回内积结果，dim 指定计算的维度
dot(A,B)
```

```
ans = 70
```

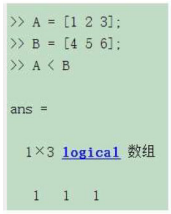
```
dot(A,B,1)
```

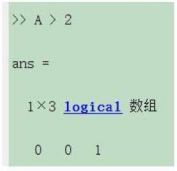
```
ans = 1×4
     5    12    21    32
```

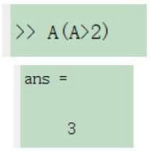
**关系运算**

6种数组关系运算符：小于<、小于等于<=、大于>、大于等于>=、恒等于==、不等于~=

运算规则：
- ✓ 当两个标量比较时，直接比较两个数大小，返回0或1
- ✓ 当两个维度相等的数组比较时，逐一比较两个数组相同位置元素，返回一个与输入数组维度相同的结果数组




思考：这个式子的结果？

18

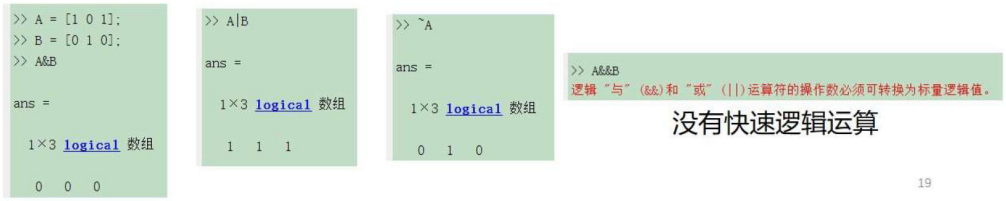
```
A = [1,2,3];
B = [4,5,6];
A<B
```

```
ans = 1×3 logical 数□
   1   1   1
```

**逻辑运算**

3种逻辑运算符，与(&)或(|)非(~)，运算规则如下：

- 非零元素为真（1表示），零元素为假（0表示）
- 两个比较量是维度相等的数组时，注意比较两个位置元素，给出比较结果，返回与输入数组维度相同的逻辑数组（元素为0/1）




```
A = [1,0,1];
B = [0,1,0];
A&B
```

```
ans = 1×3 logical 数□
   0   0   0
```

```
A|B
```

```
ans = 1×3 logical 数□
   1   1   1
```

```
~A
```

```
ans = 1×3 logical 数□
   0   1   0
```

## 字符串

```
% 单引号和双引号的区别
'a'+'b'
```

```
ans = 195
```

```
"a"+"b"
```

```
ans =
"ab"
```

```
% 数据类型不同 -- 同样的运算符处理结果也不同

% 拼接
s1='we';
s2='do';
s3='not';
s4='have';
s5='example';
s7=[s1,' ',s2,' ',s3,' ',s4,' ',s5]
```

```
s7 =
'we do not have example'
```

```
% 逻辑运算
s5 == "example"
```

```
ans = logical
   1
```

```
s5 == 'e'
```

```
ans = 1×7 logical 数□
   1   0   0   0   0   0   1
```

```
% 借助位置进行元素修改
s5(s5=='e') = 'Z';
s5
```

```
s5 =
'ZxamplZ'
```

- 可以进行逻辑操作

```
>> s1 = 'we';
>> s2 = 'do';
>> s3 = 'not';
>> s4 = 'have';
>> s5 = 'example';
```

```
>> s4 == 'e'

ans =

  1×4 logical 数组

   0   0   0   1
```

```
>> s5(s4 == 'e') = 'X'
```

```
>> s5(s4 == 'e') = 'X'

s5 =

    'exaXple'
```

思考该式的输出结果

- MATLAB中的字符串也可以使用数组的思维进行理解，利用下标进行访问和修改
- 注意“ ”与‘ ’的区别，引起数据类型的差异
- 能够进行逻辑运算，获得对应的下标进行运算/修改

```
student(1).name = 'Rita';
student(1).gender = 'F';
student(1).ID = 21312222;
student(1).major = 'CS';
student(1).grade = [100,99,98];

student(2).name = 'Sam';
student(2).gender = 'M';
student(2).ID = 21313333;
student(2).major = 'CC';
student(2).grade = [97,99,98];
student
```

```
student = 包含以下字段的 struct
```

| 字段 | name | gender | ID | major | grade | other_info |
|---|---|---|---|---|---|---|
| 1 | 'Rita' | 'F' | 21312222 | 'CS' | [100,99,... | [ ] |
| 2 | 'Sam' | 'M' | 21313333 | 'CC' | [97,99,98] | 1×1 struct |

- 结构体中存放数据的部分可称为“域”，能够存放**任何类型、任何大小**的数组和字符串等数据；使用点号.进行创建和访问
- 不同结构的同名域可以存放不同(类型)内容（内存地址不同）
- 结构体数组可以有一维、二维、更高维，但一维用的最多

```
% 一个结构体中某一元素为结构体
my_student.name = 'Sam';
my_student.gender = 'M';
my_student.ID = 21313333;
my_student.major = 'CC';
my_student.grade = [97,99,98];
my_student.other_info.idx1=1;
my_student.other_info.idx2=3;
my_student
```

```
my_student = 包含以下字段的 struct:
          name: 'Sam'
        gender: 'M'
            ID: 21313333
         major: 'CC'
         grade: [97 99 98]
    other_info: [1×1 struct]
```

- MATLAB中提供了struct函数用于创建结构数组，调用格式为：

s = struct('field1', values1, 'field2', values2, ...)，其中field表示字段名，value表示对应的字段值

s = struct('field1', {}, 'field2', {}, ...)，用指定字段field1、field2建立一个空结构体（没有数据）

s = struct([])，建立一个没有字段的空结构

s = struct(obj)，讲对象obj转换为它的等价结构体

```
field = 'test1';
values = {'123',[4,5,6]};
s = struct(field,values);
s
```

```
s = 包含以下字段的 struct
```

| 字段 | test1 |
|---|---|
| 1 | '123' |
| 2 | [4,5,6] |

```
s.test1
```

```
ans =
'123'
ans = 1×3
     4     5     6
```

## 可用的函数操作

| 函数 | 说明 |
| --- | --- |
| deal | 将输入分配给输出 |
| isfield | 测试是否是结构数组的字段 |
| struct2cell | 将结构数组转换为单元数组 |
| fieldnames | 得到结构的字段名 |
| isstruct | 是否为结构 （1-是； 0-否） |
| struct | 建立结构数组 |

注意值的问题

思考该式的输出

- 示例

```
>> C = {"A", "B", "C"};
>> [a, b, c] = deal(C{:})
```

```
a =
    "A"
```

```
b =
    "B"
```

```
c =
    "C"
```

```
C = {"A","B","C"};
[a,b,c] = deal(C{:}) % 将输入分配给输出
```

```
a =
"A"
b =
"B"
c =
"C"
```

## 其他函数操作

| | |
| --- | --- |
| cell2struct | Convert cell array to structure array |
| fieldnames | Field names of structure, or public fields of object |
| getfield | Field of structure array |
| isfield | Determine whether input is structure array field |
| isstruct | Determine whether input is structure array |
| orderfields | Order fields of structure array |
| rmfield | Remove fields from structure |
| setfield | Assign values to structure array field |
| struct | Create structure array |
| struct2cell | Convert structure to cell array |
| structfun | Apply function to each field of scalar structure |