



第6章 MATLAB数据结构



- 6.1 多维数组
- 6.2 结构体
- 6.3 细胞
- 6.4 字符串
- 6.5 本章小结



- **6.1 多维数组**
- 6.2 结构体
- 6.3 细胞
- 6.4 字符串
- 6.5 本章小结



6.1.1 多维数组的表现形式

在**Matlab**中习惯上将二维以上的数组称为多维数组，二维数组中的第一维称为“行”第二维称为“列”，而对于三维数组的第三位则是习惯性的称为“页”。

在**Matlab**中将三维及三维以上的数组统称为高维数组，三维数组也是高级运算的基础，下面将介绍几种创建三维数组**Matlab**语句。



6.1.2 多维数组的创建

多维数组的建立与矩阵的建立方法类似，常用的多维数组建立的方式主要有以下四种方式：

- (1) 利用下标建立多维数组。
- (2) 利用**MATLAB**数产生多维数组。
- (3) 利用**cat**函数建立多维数组。
- (4) 用户自己编写**M**文件产生多维数组，即用户自己编写代码产生多维数组。

1、利用下标建立多维数组


在**MATLAB**中，多维数组可以通过先建立二维矩阵，然后再将其扩展为相应的多维数组。

例如，首先在命令行窗口利用**MATLAB**语句产生一个 3×3 的方阵**A**，其代码如下：

```
A=[5 7 2; 0 1 2; 3 4 2];
```

A实际上也可看作是 $3 \times 3 \times 1$ 的数组，扩展其维数，将原始的**A**作为第一个维度的数据，对第2个维度进行赋值，可得

```
A(:, :, 2)=[2 5 3; 4 2 8; 2 0 3]
```



MATLAB的运行结果如下:

A(:,:,1) =

5	7	2
0	1	2
3	4	2

A(:,:,2) =

2	5	3
4	2	8
2	0	3

这说明我们已经产生了一个 **$3 \times 3 \times 2$** 的多维矩阵**A**。

如果要扩展维的所有元素均相同, 则可用标量来输入。例如:

A(:,:,3)=6;

A(:,:,3)



MATLAB的运行结果如下：

ans =

```
6    6    6
6    6    6
6    6    6
```

进一步扩展维数可得到**4**维数组：

```
A(:,:,1,2)=eye(3);
A(:,:,2,2)=5*eye(3);
A(:,:,3,2)=10*eye(3);
size(A)
```

MATLAB的运行结果如下：

ans =

```
3    3    3    2
```

这说明得到的矩阵
A为 **$3 \times 3 \times 3 \times 2$** 维。



2、利用MATLAB函数产生多维数组

利用**MATLAB**的函数(如**rand**、**randn**、**ones**、**zeros**等)也可直接产生多维数组，在函数调用时可指定每一维的尺寸。

例如 为产生 $10 \times 3 \times 2$ 维的正态分布随机数**A**，可输入代码：

A=randn(10, 3, 2)

为产生各元素相同的多维数组，可采用**ones**函数，也可采用**repmat**函数，可输入代码：

B=5*ones(3, 4, 2);

C=repmat(5, [3 4 2]);



3、利用cat函数建立多维数组

任何两个维数适当的数组可利用**cat**函数按指定维进行连接，从而可以组合这两个数组产生更高维的数组。例如，输入代码：

```
A=[2 8; 0 5]; B=[1 8; 2 4];
```

当它们沿着第三维以上的维进行连接时，可得到多维数组，输入代码：

```
C=cat(3,A,B);
```


```
D=cat(4,A,B);
```

```
size(C)
```

MATLAB的运行结果如下：

```
ans =
```

```
2 2 2
```



然后用如下的代码查看**D**的大小：
size(D)

MATLAB的运行结果如下：

ans =
2 2 1 2

4、用户自定义**M**文件产生多维数组

对于任意指定的多维数组，用户都可以编写专门的**M**文件来产生，这样可避免在设计中过多地在程序中输入数据。



6.1.2 多维数组的转换

在建立多维数组之后改变其尺寸和维数，改变的方法大概有两种：

一、是直接多维数组中添加或删除元素，从而改变多维数组的维度或改变每个维度的数据量；

二、是利用**reshape**函数，在保持所有元素个数和内容不变的前提下，改变多维数组每一维度的尺寸和多维数组的维数。



例1 使用**cat**函数将两个 3×4 的随机矩阵进行连接，形成一个 $3 \times 4 \times 2$ 的多维数组。

MATLAB代码:

```
M=cat(3,fix(15*rand(3,4)),fix(10*rand(3,4)))
```

MATLAB的运行结果如下:

M(:,:,1) =

2	9	0	13
3	4	11	6
2	2	6	6

M(:,:,2) =

8	6	6	5
5	8	3	7
2	0	8	4

例2 将例6-3生产的矩阵**M**变成**4×6**的矩阵。

MATLAB代码如下：

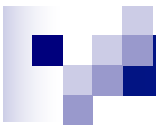
N=reshape(M,4,6)

MATLAB的运行结果如下：

N=

2	4	6	8	8	8
3	2	13	5	0	5
2	0	6	2	6	7
9	11	6	6	3	4

可以看出，**reshape**函数是按列方式操作的。



- 6.1 多维数组
- **6.2 结构体**
- 6.3 细胞
- 6.4 字符串
- 6.5 本章小结



6.2.1 结构体构造和赋值

MATLAB建立结构体有两种方式:

- (1) 使用**struct**函数。
- (2) 使用赋值语句。

1. 利用**struct**函数建立结构阵列

struct的使用格式为:

```
s = struct('field1',values1,'field2',values2,...);
```

%%注意引号

该函数将生成一个具有指定字段名**field**和相应数据**values**的结构数组，其中**field1**为字段1，**values1**为其对应的值，**field2**为字段2，**values2**为其对应的




struct的使用格式为:

s = struct('field1',values1,'field2',values2,...);

需要注意的是数据**values1**、**values2**等必须为具有相同维数的数据。并且数据**values1**、**values2**等可以是细胞数组、结构体等，每个**values**的数据被赋值给相应的**field**字段。

当**values**为细胞数组的时候，生成的结构数组的维数与细胞数组的维数相同。而在数据中不包含元胞的时候，得到的结构数组的维数是**1×1**的。




例 将一个温室的数据最少包括这样几个字段：
“温室名称” “温室大小” “温室温度” “温室
种植物”，在**MATLAB**中利用函数**struct**，建立
温室群的数据库。

在**MATLAB**中建立结构体的代码如下：

```
name='六号房';  
volume='3200立方米';  
temperature = '28';  
plant = 'cabbage';  
green_house_6 =  
struct('name',name,'volume',volume,'temperature',  
temperature,'plant',plant);  
green_house_6
```

MATLAB的运行结果如下：

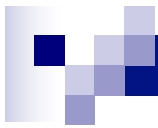
```
green_house_6 =  
    name: '六号房'  
  volume: '3200立方米'  
temperature: '28'  
   plant: 'cabbage'
```



当然，如果我们建立的温室不止一个，这个时候就可以用结构体数组来将所有的温室数据集合到一起了。例如，我们现在有**6个温室**，其中除了名字不一样外，温室的大小、温度和种植物都一样，那么现在温室的数据库的建立如下：

```
volume='3200立方米';  
temperature = '28';  
plant = 'cabbage';  
for i = 1:6  
    name = [num2str(i) '号房'];  
    green_house(i)=struct('name',name,'volume',  
        volume,'temperature',temperature,'plant',plant);  
end  
green_house
```

MATLAB的运行结果：
green_house =
1x6 struct array with fields:
 name
 volume
 temperature
 plant



在**MATLAB**中查看**green_houses**数组某个元素的值:

green_house(2)

MATLAB的运行结果如下:

ans =

name: '2号房'

volume: '3200立方米'

temperature: '28'

plant: 'cabbage '



2. 利用赋值语句建立结构阵列

MATLAB也可以利用赋值语句对结构阵列的各个字段进行赋值，注意结构名与域名字段之间用句点分隔，下面举例说明在**MATLAB**中如何利用赋值语句构造结构数组。

例 一个病人的数据最少包括这样几个字段：“病人姓名”“治疗的费用”“检查的参数”，在**MATLAB**中利用赋值语句建立病人的数据库。

在**MATLAB**建立结构体的代码如下：



```
clear
```

```
patient1.name = 'John Doe';
```

```
patient1.billing = 127;
```

```
patient1.test = [79 75 73; 180 178 177.5; 220 210 205];
```

```
patient1
```


MATLAB的运行结果如下:

```
patient1 =
```

```
    name: 'John Doe'
```

```
    billing: 127
```

```
    test: [3x3 double]
```



当然，也可以利用细胞数组一次输入多个结构元素，
即可以输入多个病人的情况，其**MATLAB**代码为：

```
clear
n={'John Doe' 'Ann Lane' 'Alan Johnson'}; %细胞数组
b=[127 28.5 95.8];
t1=[79 75 73; 180 178 177.5; 220 210 205];
t2=[68 70 68; 118 118 119; 172 170 169];
t3=[37 38 36; 119 121 120; 165 166 159];
t(:, :, 1) = t1; t(:, :, 2) = t2; t(:, :, 3) = t3;
for i = 1:3
    patient2(i).name = n{i}; patient2(i).billing = b(i);
    patient2(i).test = t(:, :, i);
end
patient2
```

6.2.2 结构体的嵌套

在**MATLAB**中，结构体的域值可以是另一个已定义过的结构，这就是结构体的嵌套使用。

先利用**struct**建立嵌套结构阵列的一个元素，然后利用赋值语句进行扩展，如输入代码：

```
A=struct('data',[3 4 7;8 0 1],'nest', ...  
struct('testnum','Test 1','xdata',[4 2 8],'ydata',[7 1 6]));  
A(2).data=[9 3 2;7 6 5];  
A(2).nest.testnum='Test 2';  
A(2).nest.xdata=[3 4 2];  
A(2).nest.ydata=[5 0 9];  
A(1).data  
A(2).nest
```

MATLAB的运行结果如下：

```
ans =  
     3     4     7  
     8     0     1  
ans =  
testnum: 'Test 2'  
xdata: [3 4 2]  
ydata: [5 0 9]
```




- 6.1 多维数组
- 6.2 结构体
- **6.3 细胞**
- 6.4 字符串
- 6.5 本章小结




6.3.1 细胞数组的创建

细胞数组的创建主要有函数法和直接赋值法。函数法是指使用**MATLAB**提供的**cell**（）函数创建细胞数组。直接赋值法是指直接在命令行中给细胞数组的每个元素赋值，或者实验大括号“**{}**”创建细胞数组。

1、利用函数创建细胞数组

在**MATLAB**中可以利用**cell()**函数生产一个细胞数组，这个函数先对细胞数组中的元素进行内存空间的预分配，然后再赋值。

该函数的调用格式为：



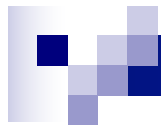
A=cell(n): 生成 $n \times n$ 的细胞数组**A**。

A=cell(m, n)或者**A=cell([m, n])**: 生成 $m \times n$ 的细胞数组**A**。

A=cell(m, n, p, ...)或者**A=cell([m, n, p, ...])**: 生成 $m \times n \times p \times \dots$ 的细胞数组**A**。

A=cell(size(B)): 生成一个与数据**B**具有相同大小的细胞数组**A**。

例 设一个彩色图像的**R**通道为 256×256 的全**1**矩阵，**G**通道为 256×256 的全**0**矩阵，**B**通道为 256×256 的单位矩阵，则利用细胞数组函数法表示这个彩色图像的代码如下：



clear

% R、G、B单通道定义的值

R = ones(256, 256);

G = zeros(256, 256);

B = eye(256, 256);

% 定义细胞数组

Image = cell(3, 1);

% 对细胞数组赋值

Image{1} = R;

Image{2} = G;

Image{3} = B;

Image

Image =

[256x256 double]

[256x256 double]


[256x256 double]



2、直接赋值法

细胞数组的直接赋值可以使用小括号表示细胞数组的下标，而细胞中的内容则需要使用大括号“{}”括起来。如果使用大括号括起细胞的下标时，细胞中的内容无须另加标点，与一般数组的元素输入相同。这两种方法的效果是一样的，但要注意符号前后的配合。

使用大括号“{}”创建细胞数组的方法类似于使用中括号“[]”生成一般的数组，行之间元素用分号“;”分割，列之间元素用逗号“,”或者空格分割。



例 设一个彩色图像的R通道为 256×256 的全1矩阵，G通道为 256×256 的全0矩阵，B通道为 256×256 的单位矩阵，则利用直接法创建细胞数组表示这个彩色图像的代码如下：

```
clear  
% R、G、B单通道定义的值  
R = ones(256, 256);  
G = zeros(256, 256);  
B = eye(256, 256);  
% 使用直接法创建细胞数组  
Image{1} = R;  
Image{2} = G;  
Image{3} = B;  
Image
```

6.3.2 细胞数组的访问

MATLAB提供了两种方式对细胞数组的进行访问，其中大括号访问的是细胞数组中细胞的内容，可以对细胞中的内容进行进一步的操作，而小括号访问的是细胞数组的元胞，是个整体，无法对细胞中的具体数据进行操作。

例 利用大括号访问细胞数组的代码如下：

```
a={20,'matlab',ones(2,3)}
```

则**MATLAB**的运行结果如下：

```
a =
```

```
[20] 'matlab' [2x3 double]
```



查看**cell**数组**a**中的第三个**cell**的代码如下：

a(3)

则**MATLAB**的运行结果如下：

ans =

[2x3 double]

查看**cell**数组**a**中第三个**cell**的类型的代码如下：

class(a(3))

则**MATLAB**的运行结果如下：

ans =

cell



查看**cell**数组**a**中第三个**cell**的内容的代码如下:

a{3}

则**MATLAB**的运行结果如下:

ans =

```
1    1    1
1    1    1
```

删除**cell**数组**a**中的第三个**cell**的代码:

a(3)=[]

则**MATLAB**的运行结果如下:

a =

```
[20] 'matlab'
```



6.3.2 细胞数组的显示

在**MATLAB**中利用**celldisp()**和**cellplot()**函数显示细胞数组，其中**celldisp()**函数可以显示细胞数组的具体内容，而**cellplot()**函数则以图形的方式显示细胞数组。

1、 **celldisp()**函数的调用格式为：

celldisp (A): 显示细胞数组**A**中的具体内容。

celldisp (A, 'name'): 以字符串**name**细胞数组的名称，然后显示细胞数组**A**中的具体内容。



例 利用**celldisp()**显示细胞数组的内容:

```
A={20,'matlab',ones(2,3)};  
celldisp(A)
```

则**MATLAB**的运行结果如下:

A{1} =

20


A{2} =

matlab

A{3} =

1 1 1

1 1 1



以**s**为细胞数组的名称，然后显示**A**中的具体内容的代码如下：

celldisp(A, 's')

MATLAB的运行结果如下：

s{1} =

20

s{2} =

matlab

s{3} =

1 1 1

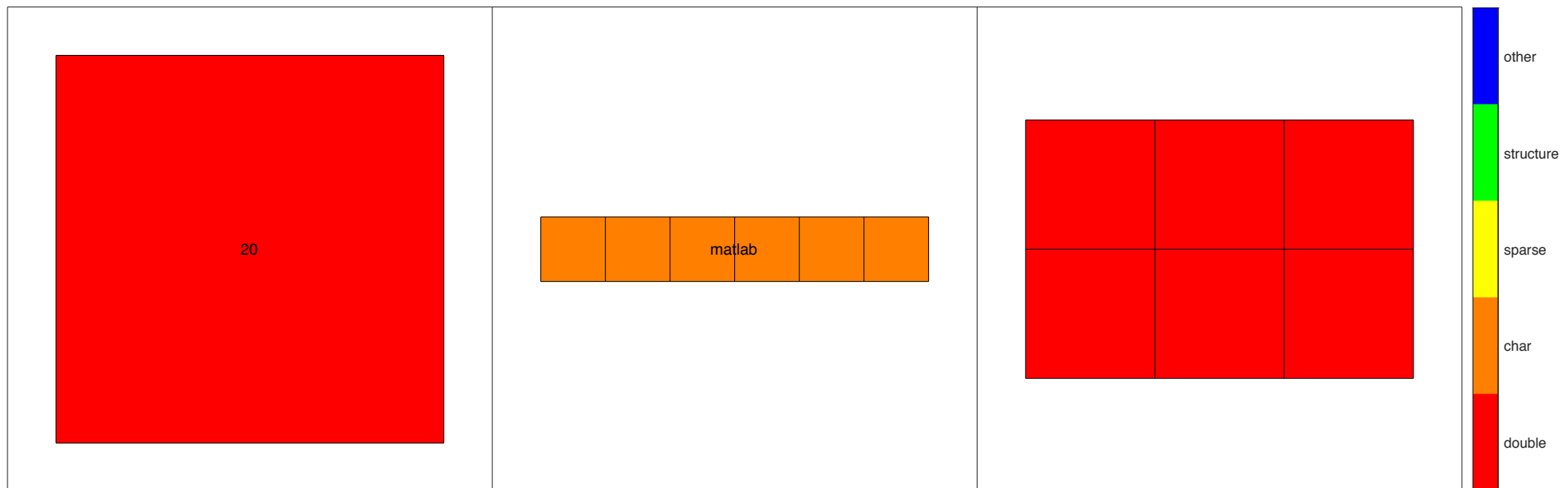
1 1 1

可以看到细胞数组在显示的时候标记了另外一个名称**s**，注意，这并不是重新建立或拷贝了一个新的细胞数组，而仅仅是用**s**去显示**A**中的内容。

2、cellplot()函数的调用格式为:

cellplot (A): 以图形化的形式显示细胞数组A。

cellplot (A, 'legend'): 以图形化的形式显示细胞数组A，同时显示不同数据类型的颜色图例标注。





matlab struct array 和 cell array的区别

- **struct**是具有属性名的，也就是**field**，所以对每个**struct**根据这个**field** 来获取内容。
- **cell**没有属性名，通过**1, 2, 3** 这样的**index**来获得相应的元素。

cell2struct

struct2cell

从内存分配上来讲，在赋值使用之前结构体可以精确地分配内存。如果你所需要使用的数据类型是固定的，并且向量或者矩阵的维数的固定的，那么推荐使用**struct**。

在**matlab**里面**cell**的适用性非常灵活，**cell**数组的每个元素的在使用过程中它的数据类型，数据尺寸都可以随意改变，但是这么方便这么好用的东西肯定也是要付出代价的。首先**cell**数组因为其强大的数据灵活性所以它的内存分配是动态的，这一定程度上会影响程序的执行效率。其次因为它赋值的极大灵活性，就算代码里面有错误例如不同数据类型之间的变量赋值，或者赋值过程中超出矩阵维度，**matlab**并不会报错，这样在**debug**方面就造成不便



- 6.1 多维数组
- 6.2 结构体
- 6.3 细胞
- **6.4 字符串**
- 6.5 本章小结



6.4.1 字符串构造

字符串的生成主要通过直接赋值法、已构成字符串类型的数据。字符串类型的数据每个字符占**2**字节的存储空间。

例 定义一个一维字符串。

MATLAB代码如下：

```
name='中国科学院心理研究所';  
name
```

则**MATLAB**的运行结果如下：

```
name =  
中国科学院心理研究所
```




可以利用**class**命令检查**name**的类型:

class(name)

则**MATLAB**的运行结果如下:

ans =

char

可以看到变量**name**的类型为字符型，再利用如下代码检测**name**数组的大小:

size(name)

则**MATLAB**的运行结果如下:

ans =

1 10



6.4.1 字符串函数


在**MATLAB**中为用户内置了丰富的字符串函数以方便的进行字符串处理，主要包括字符串一般函数、字符串的判断、字符串的连接和比较、字符串的搜索与取代、字符串与数值之间的转换。

1、字符串的一般处理函数：

(1) **char()**: 建立字符阵列，调用格式为：

S=char(X): 将**X**中以字符**ASCII**码表示的值转换成相应的字符，利用**double()**函数可作相反的变换。

S=char(C): 将单元阵列中的字符串变换成字符阵列**S**，利用**cellstr()**可作相反的变换，**C**表示字符串的单元阵列。



(2) **double ()**: 将字符阵列变换成双精度数值，其调用格式为：

Y=double(X): 可将字符阵列**X**转换成其**ASCII**码，如果**X**本身已经是双精度数值，则**double**函数不起作用。

例 举例说明**double**函数的用法。

```
s=char(reshape(32:127,32,3)');
```


```
s1='ABC';
```

```
y=double(s)
```

```
y1=double(s1)
```

```
y  
[3x32] double
```

```
y1 =  
65 66 67
```

- 
- (3) **cellstr()**: 从字符阵列中建立细胞数组。
 - (4) **blanks()**: 建立空格字符。
 - (5) **deblank()**: 删除字符串末尾的空格。
 - (6) **upper()**: 将字符串变成大写。
 - (7) **lower()**: 将字符串变成小写。



2、字符串的判断函数

(1) **ischar ()**: 检测到字符阵列(字符串)时为逻辑真, 调用格式为:

k=ischar(s): 当**s**为字符阵列或字符串时, **k**为逻辑真(其值为1), 否则**k**为0。

例 判断**s='University'**是否含有字符串。

s='University';

ischar(s)

则**MATLAB**的运行结果如下:

ans =

1



(2) **iscellstr ()**: 检测到字符串的细胞数组时为逻辑真，用法与**ischar()**一样。

(3) **isletter ()**: 检测到英文字母时为逻辑真，调用格式为:

blsL=isletter(str): 当**str**中某一位为英文字母时，对应的**blsL**中的元素为逻辑真，否则为逻辑假。

例 判断**str='Min is 1200'**那些是字母。


str='Min is 1200';

blsL=isletter(str)

则**MATLAB**的运行结果如下:

blsL =

1 1 1 0 1 1 0 0 0 0 0



(4) **isspace ()**: 检测到空格时为逻辑真，其调用格式为:

blsL= isspace (str): 当**str**中某一位为空白(即空格、换行、回车、制表符**Tab**、垂直制表符、打印机走纸符)时，相应的**TF**中的单元为逻辑真，否则为逻辑假。

例 判断**str='Min is 1200'**那些是空格。

str='Min is 1200';

blsL= isspace (str)

则**MATLAB**的运行结果如下:

blsL =

0 0 0 1 0 0 1 0 0 0 0

3、字符串连接和比较函数：

(1) **strcat ()**：字符串连接函数，其调用格式为：

t=strcat(s1, s2, s3, ...)：可按水平方向连接字符串**s1, s2, s3, ...**，并忽略尾部添加的空格。所有的输入**s1, s2, s3, ...**必须具有相同的行数。当输入全为字符阵列时，**t**也为字符阵；当输入中包含有字符串的单元阵列时，则**t**为单元阵列。当**s**为字符阵列或字符串时，**k**为逻辑真(其值为1)，否则**k**为0。

例 连接**s1='Psychology', s2=' Institute'**
s1='Psychology';
s2=' Institute';
t=strcat(s1,s2)

则MATLAB的运行结果如下：
t =
Psychology Institute




(2) **strvcat ()**: 字符串的垂直连接。

t=strvcat(s1, s2, s3, ...): **strvcat**与**strcat**函数类似，只是按垂直方向连接字符串**s1, s2, s3, ...**，即以**s1, s2, s3, ...**作为**t**的行，为此会自动在**s1, s2, s3, ...**的尾部补空格以形成字符串矩阵。

(3) **strcmp ()**: 字符串的比较函数。

k=strcmp(s1, s2): 可对两个字符串**s1**和**s2**进行比较，如果两者相同，则返回逻辑真(其值为**1**)，否则返回逻辑假(**0**)。

TF=strcmp(S, T): **S**、**T**为字符串单元阵列，**TF**与**S**、**T**尺寸相同，且当相应**T**、**S**元素相同时其值为**1**，否则为**0**。注意**T**、**S**必须具有相同的尺寸，或者其中之一为标量。



(4) **strcmpi ()**: 字符串的比较函数, 在比较时忽略字母的大、小写, 用法与**strcmp()**函数类似, 这里不做赘述。

(5) **strncmp ()**: 比较两个字符串的前**n**个字符是否相同。

k= strncmp (s1, s2, n): 可对两个字符串**s1**和**s2**的进行比较, 如果两者相同, 则返回逻辑真(其值为**1**), 否则返回逻辑假(**0**)。

TF= strncmp (S, T, n): **S**、**T**为字符串单元阵列, **TF**与**S**、**T**尺寸相同, 且当相应**T**、**S**元素前**n**个字符相同时其值为**1**, 否则为**0**。注意**T**、**S**必须具有相同的尺寸, 或者其中之一为标量。



例6 比较s1="Hebei", s2=' Hebei University '这两个字符串前5个字符是否相同。

s1='Hebei';

s2='Hebei University';

if strncmp(s1,s2, 5)

disp('字符串相等!')

else

disp('字符串不相等!')

end

则MATLAB的运行结果如下：

字符串相等!

4、字符串的搜索与取代函数

(1) **findstr ()**: 在长的字符串中查找短的子字符串。

k= strncmp (s1, s2): 如果**s1**比**s2**长的话, 则在**s1**中搜索**s2**, 如果**s2**比**s1**长的话, 则在**s2**中搜索**s1**, **k**表示短字符串在**s1**中的位置。

例 搜索**s1='e'**在**s2=' Hebei University '**中的位置。

s1='e';

s2='Hebei University';

k= findstr(s1, s2)

则**MATLAB**的运行结果如下:

k =

2 4 11



(2) **strfind (s1, s2)**: 在**s1**中找**s2**, 如果**s2**比**s1**长, 则返回空。如果将上面的例子改成:

k= strfind(s2, s1)

k= strfind(s1, s2)


则**MATLAB**的运行结果如下:

k =

2 4 11

k =

[]



(3) **strjust()**: 调整字符阵列。

t=strjust(s), t=strjust(s, 'left'): 按左对齐排列。

t=strjust(s, 'center'): 按居中对齐排列。

t=strjust(s, right): 按右对齐排列。

(4) **strmatch()**: 查找匹配字符串。

k=strmatch(s1, s2): 可在**s2**字符串中找出以**s1**开头的字符串位置**k**。

k=strmatch(s1, s2, 'exact'): 可在**s2**字符串中找出严格以**s1**开头的字符串的位置**k**。

(5) **strrep ()**: 字符串的搜索与取代。

str=strrep(s1, s2, s3): 可在字符串**s1**中找出子字符串**s2**, 并以**s3**取代。



例 查找字符并进行替换

```
s1 = 'Hebei University';
```

```
s2 = 'Hebei';
```

```
s3 = 'Beijing';
```

```
str=strrep(s1, s2, s3)
```

则**MATLAB**的运行结果如下：

```
str =
```

```
Beijing University
```



(5) **strtok ()**: 找出字符串的首部。

token=strtok(str, delimiter): 可找出字符串**str**的首部，即位于第一个分隔符**delimiter**之前的一串字符，其中**delimiter**用于指定有效的分隔符。

token=strtok(str)可以指定采用缺省的分隔符，即空格(**ASCII**码为32)、**Tabs**(**ASCII**码为9)和回车(**ASCII**码为13)。

[token, rem]=strtok(...)除得到字符串的首部**token**外，还得到了剩余字符串**rem**。



5、字符串与数值之间的转换函数：

(1) **eval ()**: 计算以字符串表示的**MATLAB**表达式。

a=eval(express): 可计算出**MATLAB**表达式
expression的值, **expression = [string1,**
int2str(var), string2, ...]。

[a1, a2, a3, ...] = eval(function(b1, b2, b3, ...)):
b1, b2, b3, ...为函数**function**的输入变量, 计算结果保存在**a1, a2, a3, ...**中。



(2) **num2str ()**: 将数值变换成字符串。

T=num2str(X): 可将矩阵**X**变换成以四位小数精度表示的字符串，如需要可指定以指数形式表示。

T=num2str(X, N): 以**N**位有效数字将矩阵**X**变换成字符串。

T=num2str(X, format): 使用指定的格式**format**将矩阵**X**变换成字符串。

例 将**pi**和**eps**转换为字符串。

num2str(pi)

num2str(eps)

MATLAB的运行结果如下:

ans =

3.1416

ans =

2.2204e-016



(3) **int2str ()**: 将整数变换成字符串。

T=int2str(X): 可将整数**X**变换成字符串，输入**X**可以是标量、向量，还可以是矩阵，对输入的非整数值在变换之前被截断。

(4) **str2num ()**: 将字符串变换为数值。

n=str2num (s): **str2num**是**num2str**的逆函数，可将表示数值的字符串**s**变换为数值**n**。



读写Excel文件

examp7_1_1.xls中的数据。

	A	B	C	D	E	F	G	H
1	序号	班名	学号	姓名	平时成绩	期末成绩	总成绩	备注
2	1	60101	6010101	陈亮	0	63	63	
3	2	60101	6010102	李旭	0	73	73	
4	3	60101	6010103	刘鹏飞	0	0	0	缺考
5	4	60101	6010104	任时迁	0	82	82	
6	5	60101	6010105	苏宏宇	0	80	80	
7	6	60101	6010106	王海涛	0	70	70	
8	7	60101	6010107	王洋	0	88	88	
9	8	60101	6010108	徐靖磊	0	80	80	
10	9	60101	6010109	阎世杰	0	92	92	
11	10	60101	6010110	姚前树	0	84	84	
12	11	60101	6010111	张金铭	0	95	95	
13	12	60101	6010112	朱星宇	0	82	82	
14	13	60101	6010113	韩宏洁	0	75	75	
15	14	60101	6010114	刘菲	0	71	71	
16	15	60101	6010115	苗艳红	0	70	70	
17	16	60101	6010116	宋佳艺	0	80	80	
18	17	60101	6010117	王峰瑶	0	78	78	

数据导入工具 Import data

IMPORT

VIEW

Range: A1:H52

Output Type: Table

Variable Names Row: 1

Text Optio...

UNIMPORTABLE CELLS

Import Selection

SELECTION

IMPORTED DATA

IMPORT

examp7_1_1.xls

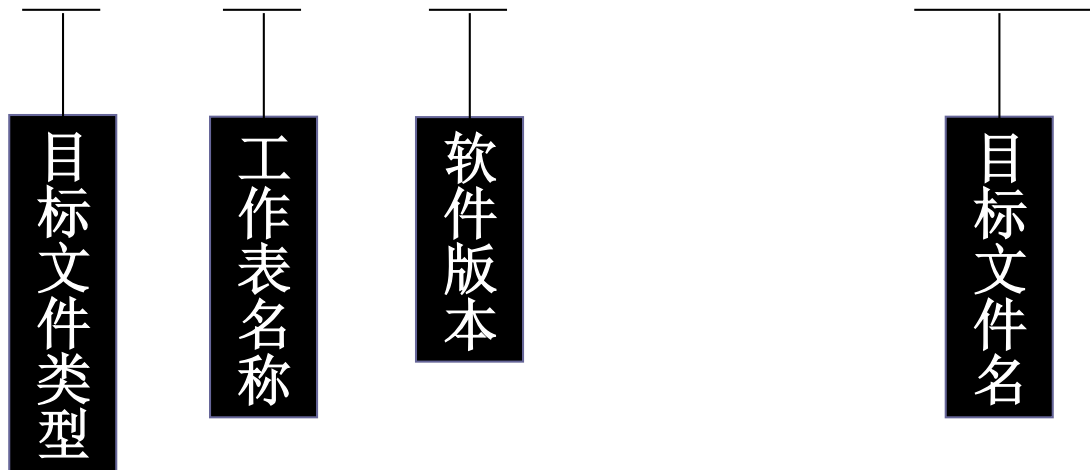
	A	B	C	D	E	F	G	H
	examp711							
	VarName1	VarNam...	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8
	Number	Number	Number	Text	Number	Number	Number	Text
1	序号	NaN	学号	姓名	平时成绩	期末成绩	总成绩	备注
2	1	60101	6010101	陈亮	0	63	63	
3	2	60101	6010102	李旭	0	73	73	
4	3	60101	6010103	刘鹏飞	0	0	0	缺考
5	4	60101	6010104	任时迁	0	82	82	
6	5	60101	6010105	苏宏宇	0	80	80	
7	6	60101	6010106	王海涛	0	70	70	
8	7	60101	6010107	王洋	0	88	88	
9	8	60101	6010108	徐靖磊	0	80	80	
10	9	60101	6010109	阎世杰	0	92	92	
11	10	60101	6010110	姚前树	0	84	84	
12	11	60101	6010111	张金铭	0	95	95	
13	12	60101	6010112	朱星宇	0	82	82	
14	13	60101	6010113	韩宏洁	0	75	75	
15	14	60101	6010114	刘菲	0	71	71	
16	15	60101	6010115	苗艳红	0	70	70	
17	16	60101	6010116	宋佳艺	0	80	80	
18	17	60101	6010117	王峥瑶	0	78	78	
19	18	60101	6010118	肖君扬	0	80	80	
20	19	60101	6010119	徐欣露	0	69	69	
21	20	60101	6010120	杨姗姗	0	81	81	
22	21	60101	6010121	姚丽娜	0	49	49	
23	22	60101	6010122	张萌	0	91	91	
24	23	60101	6010123	张婷婷	0	76	76	
25	24	60101	6010124	褚子贞	0	76	76	

Sheet1

一、调用xlsinfo函数获取文件信息

1. xlsinfo函数调用格式

➤ `[typ, desc, fmt] = xlsinfo(filename)`



```
[typ, desc, fmt] = xlsinfo('examp7_1_1.xls')
```

```
typ =
```

```
    'Microsoft Excel Spreadsheet'
```

```
desc =
```

```
    1×3 cell array
```

```
    {'Sheet1'} {'Sheet2'} {'Sheet3'}
```

```
fmt =
```

```
    0×0 empty char array
```

二、调用xlsread函数读取数据

1. xlsread函数调用格式

➤ `[num, txt, raw] = xlsread(filename, sheet, range)`

读取的数值型数据


读取的文本数据

未经处理的元胞数组

目标文件名

工作表序号或名称

读取的单元格区域



【例7.2-2】 调用xlsread函数读取文件examp7_1_1.xls第1个工作表中区域A2:H4 的数据。

% 第一种方式:

```
>> num = xlsread('examp7_1_1.xls','A2:H4')
```

% 第二种方式:

```
>> num = xlsread('examp7_1_1.xls',1,'A2:H4')
```

% 第三种方式:

```
>> num = xlsread('examp7_1_1.xls','Sheet1','A2:H4')
```

Or: convert to CSV file and import as text



【例7.2-3】 将文件**examp7_1_1.xls**第1个工作表中A2至C3单元格中的数据加1，并读取变换后的数据。

```
>> convertdata = xlsread('examp7_1_1.xls', '', 'A2:C3', '', @setplusone1)
```

```
convertdata =
```

```
2    60102    6010102
```

```
3    60102    6010103
```

其中**setplusone1**函数的源码见**setplusone1.m**

三、调用xlswrite函数把数据写入Excel文件

1. xlsread函数调用格式

➤ `[status, message] = xlswrite(filename, M, sheet, range)`

写操作指示变量

警告或错误信息

目标文件名

写入的数据矩阵

工作表序号或名称

写入的单元格区域



【例7.2-4】 生成一个 10×10 的随机数矩阵，将它写入Excel文件 `excel.xls` 的第2个工作表的默认区域。代码保存在m文件 `CaseXlsWrite.m` 中。

```
>> X = rand(10,10);
```

```
>> [status, message] = xlswrite('excel.xls', X, 'sheet2')
```



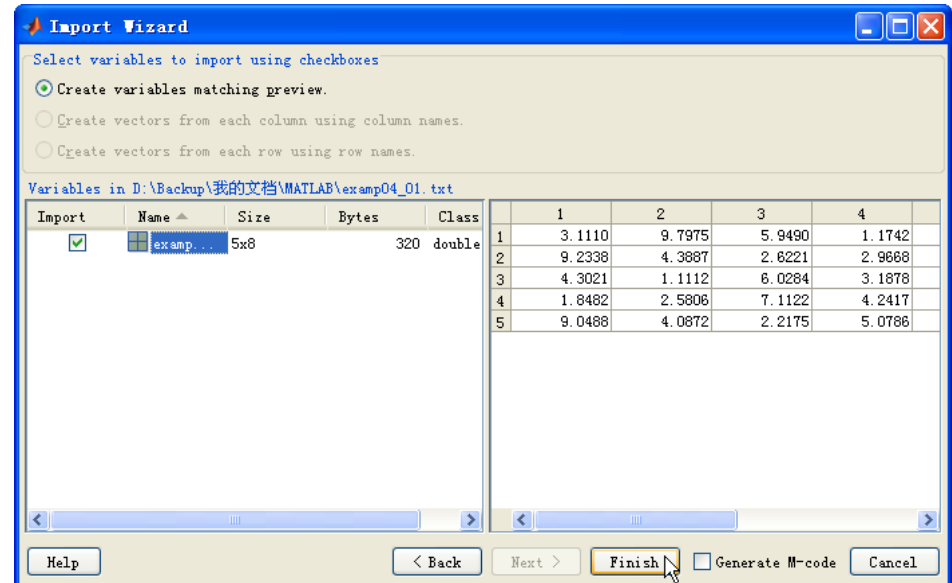
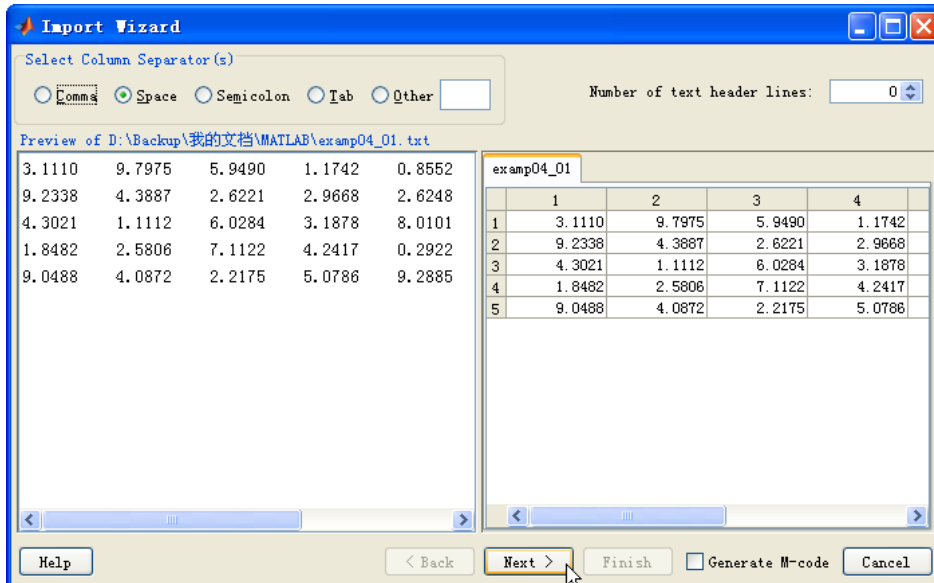
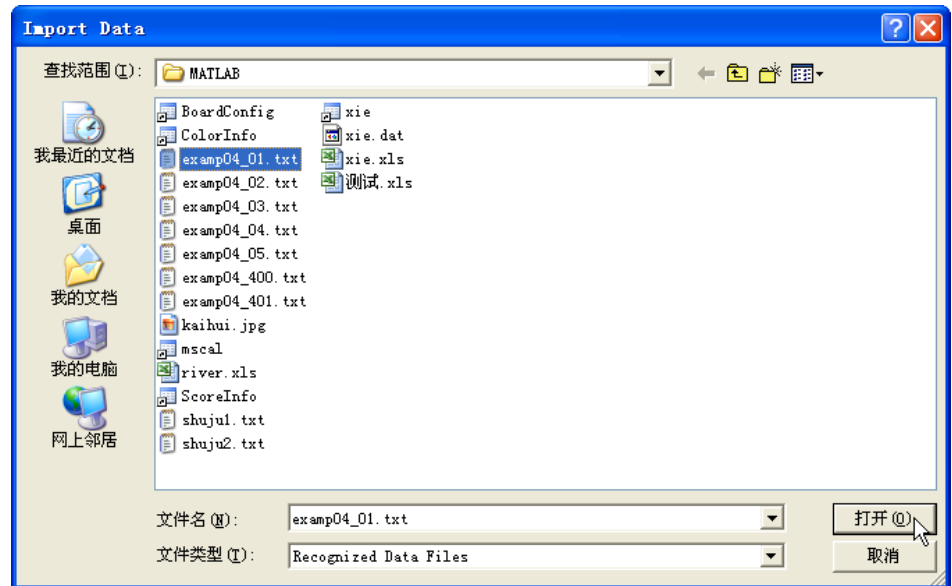
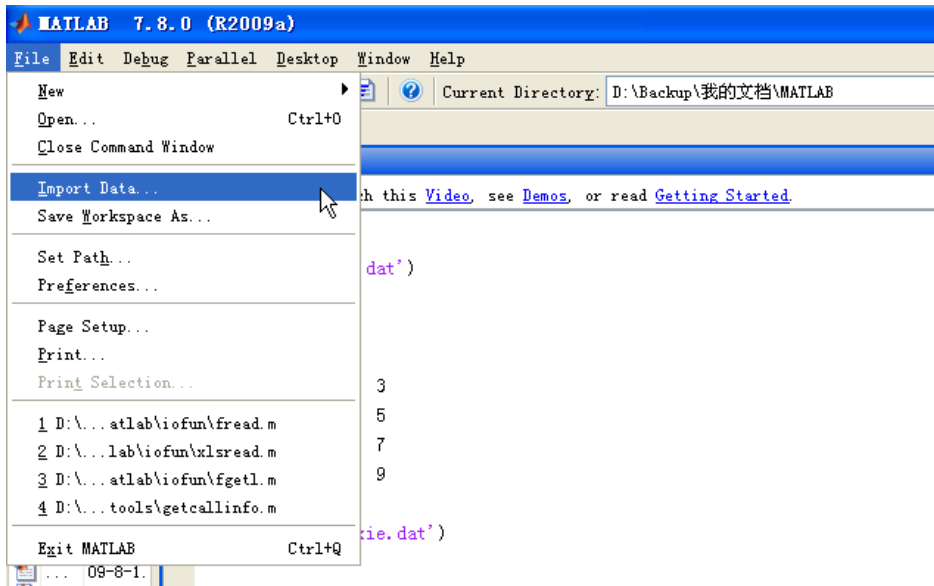
TXT文件读写数据




MATLAB中读取文本文件的常用函数

高级函数		低级函数	
函数名	说 明	函数名	说 明
load	从文本文件导入数据到 MATLAB 工作空间	fopen	打开文件，获取打开文件的信息
importdata	从文本文件或特殊格式二进制文件（如图片，avi 视频等）读取数据	fclose	关掉一个或多个打开的文件
dlmread	从文本文件中读取数据	fgets	读取文件中的下一行，包括换行符
csvread	调用了 dlmread 函数，从文本文件读取数据。过期函数，不推荐使用	fgetl	调用 fgets 函数，读取文件中的下一行，不包括换行符
textread	按指定格式从文本文件或字符串中读取数据	fscanf	按指定格式从文本文件中读取数据
strread	按指定格式从字符串中读取数据，不推荐使用此函数，推荐使用 textread 函数	textscan	按指定格式从文本文件或字符串中读取数据

一、 利用数据导入向导导入TXT文件





【例6.1-1】 利用数据导入向导读取文件`examp6_2_1.txt`至
`examp6_2_11.txt`中的数据

数据文件略去.....



二、调用高级函数读取数据

1. 调用**importdata**函数读取数据，存为**structure**

调用格式：

importdata(filename)

A = importdata(filename)

A = importdata(filename,delimiter)

A = importdata(filename,delimiter,headerline)

[A D] = importdata(...)

[A D H] = importdata(...)

[...] = importdata('-pastespecial', ...)



【例6.1-2】 调用importdata函数读取文件examp6_2_1.txt至examp6_2_11.txt中的数据

```
>> importdata('examp6_2_4.txt')
```

```
>> x = importdata('examp6_2_7.txt')
```

```
>> x = importdata('examp6_2_3.txt',',')
```

```
>> x = importdata('examp6_2_8.txt',' ',2)
```

```
>> [x, s, h] = importdata('examp6_2_7.txt')
```

```
>> FileContent = importdata('examp6_2_10.txt')
```



2. 调用load函数读取数据 (统一数据格式类型)

调用格式:

S = load(filename)

S = load(filename, variables)

S = load(filename, '-mat', variables)

S = load(filename, '-ascii')

load(...)

load ...



【例6.1-3】 调用load函数读取文件`examp6_2_1.txt`至
`examp6_2_12.txt`中的数据

```
>> load examp6_2_1.txt
```

```
>> load -ascii examp6_2_1.txt
```

```
>> x1 = load('examp6_2_2.txt')
```

```
>> x1 = load('examp6_2_2.txt', '-ascii');
```

```
>> load examp6_2_3.txt
```

```
>> load examp6_2_4.txt
```

```
.....
```



3. 调用dlmread函数读取数据

dlmread Read ASCII delimited file

调用格式:

M = dlmread(filename)


M = dlmread(filename, delimiter)

M = dlmread(filename, delimiter, R, C)

R and C specify the row R and column C where the upper-left corner of the data lies in the file. R=0 and C=0 specifies the first value in the file.

M = dlmread(filename, delimiter, range)

RANGE = [R1 C1 R2 C2] where (R1,C1) is the upper-left corner of the data to be read and (R2,C2) is the lower-right corner. RANGE can also be specified using spreadsheet notation as in RANGE = 'A1..B7'.



【例6.1-4】 调用dlmread函数读取文件examp6_2_1.txt至
examp6_2_11.txt中的数据

```
>> x = dlmread('examp6_2_3.txt')  
>> x = dlmread('examp6_2_3.txt', ',', 2, 3)  
>> x = dlmread('examp6_2_3.txt', ',', [1, 2, 2, 5])  
>> x = dlmread('examp6_2_5.txt')  
>> x = dlmread('examp6_2_6.txt')  
>> x = dlmread('examp6_2_9.txt')  
.....
```



4. 调用textread函数读取数据

调用格式:

[A,B,C,...] = textread('filename','format')

[A,B,C,...] = textread('filename','format',N)

[...] = textread(...,'param','value',...)

textread函数支持的format字符串

格式字符串	说 明	输 出
普通字符串	忽略与 format 字符串相同的内容。例如 xie%f 表示忽略字符串 xie，读取其后的浮点数	无
%d	读取一个无符号整数。例如%5d 指定读取的无符号整数的宽度为 5	双精度数组
%u	读取一个整数。例如%5u 指定读取的整数的宽度为 5	双精度数组
%f	读取一个浮点数。例如%5.2f 指定浮点数宽度为 5（小数点也算），有 2 位小数	双精度数组
%s	读取一个包含空格或其他分隔符的字符串。例如%10s 表示读取长度为 10 的字符串	字符串元胞数组
%q	读取一个双引号里的字符串，不包括引号	字符串元胞数组
%c	读取多个字符，包括空格符。例如%6c 表示读取 6 个字符	字符数组
%[...]	读取包含方括号中字符的最长字符串	字符串元胞数组
%[^...]	读取不包含方括号中字符的非空最长字符串	字符串元胞数组
%*...	忽略与*号后字符相匹配的内容。例如%*f 表示忽略浮点数	无
%w...	指定读取内容的宽度。例如%w.pf 指定浮点数宽度为 w，精度为 p	

textread函数支持的参数名与参数值列表

参数名	参数值		说 明
bufsize	正整数		设定最大字符串长度，默认值为 4095，单位是 byte.
commentstyle	matlab		忽略 % 后的内容
	shell		忽略 # 后的内容
	c		忽略 /* 和 */ 之间的内容
	c++		忽略 // 后的内容
delimiter	一个或多个字符		元素之间的分隔符。默认没有分隔符
emptyvalue	一个双精度数		设定在读取有分隔符的文件时在空白单元填入的值。默认值为 0
endofline	单个字符或 '\r\n'		设定行尾字符。默认从文件中自动识别
expchars	指数标记字符		设定科学计数法中标记指数部分的字符。默认值为 eEdD
headerlines	正整数		设定从文件开头算起需要忽略的行数
whitespace	' '	空格	把字符向量作为空格。默认值为 '\b\t'
	\b	后退	
	\n	换行	
	\r	回车	
	\t	水平 tab 键	



【例6.1-5】 调用textread函数读取文件examp6_2_1.txt至examp6_2_11.txt中的数据

```
>> x1 = textread('examp6_2_1.txt');  
>> x2 = textread('examp6_2_2.txt');  
>> x3 = textread('examp6_2_3.txt','','delimiter',',');  
>> [c1,c2,c3,c4,c5]=textread('examp6_2_4.txt','%f %f %f %f  
%f','delimiter',',;');  
>> x5 = textread('examp6_2_5.txt','', 'emptyvalue',-1)  
>> x8 = textread('examp6_2_8.txt','', 'headerlines',7)  
>> x9 = textread('examp6_2_9.txt','', 'delimiter',',', '','whitespace','+i')  
>> [c1,c2,c3,c4,c5,c6,c7,c8] = textread('examp6_2_9.txt',...  
'%f %f %f %f %f %f %f %f','delimiter',',', '','whitespace','+i');  
.....
```

三、调用低级函数读取数据

1. 调用fopen函数打开文件

调用格式:

[fid, message] = fopen(filename, permission)

[filename, permission] = fopen(fid)

permission	说 明
'rt'	以只读方式打开文件。这是默认情况
'wt'	以写入方式打开文件，若文件不存在，则创建新文件并打开。原文件内容会被清除
'at'	以写入方式打开文件或创建新文件。在原文件内容后续写新内容
'r+t'	以同时支持读、写方式打开文件
'w+t'	以同时支持读、写方式打开文件或创建新文件。原文件内容会被清除
'a+t'	以同时支持读、写方式打开文件或创建新文件。在原文件内容后续写新内容
'At'	以续写方式打开文件或创建新文件。写入过程中不自动刷新文件内容，适合于对磁带介质文件的操作
'Wt'	以写入方式打开文件或创建新文件，原文件内容会被清除。写入过程中不自动刷新文件内容，适合于对磁带介质文件的操作



2. 调用fclose函数关闭文件

调用格式:

status = fclose(fid)

status = fclose('all')



3. 调用fseek、ftell、frewind和feof函数控制读写位置

调用格式:

status = fseek(fid, offset, origin)

position = ftell(fid)

frewind(fid)

eofstat = feof(fid)



4. 调用fgets、fgetl函数读取文件的下一行

调用格式:

tline = fgets(fid)

tline = fgets(fid, nchar)

tline = fgetl(fid)



5. 调用textscan函数读取数据

调用格式:

C = textscan(fid, 'format')


C = textscan(fid, 'format', N)

C = textscan(fid, 'format', param, value, ...)

C = textscan(fid, 'format', N, param, value, ...)

C = textscan(str, ...)

[C, position] = textscan(...)

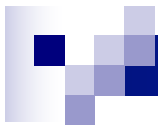


【例6.1-6】 调用textscan函数读取文件examp6_2_1.txt至
examp6_2_13.txt中的数据

```
>> fid = fopen('examp6_2_8.txt','r');  
>> fgets(fid);  
>> fgets(fid);  
>> A = textscan(fid, '%f %f %f %f %f %f', 'CollectOutput', 1)  
A =  
    [3x6 double]  
>> fgets(fid);  
>> fgets(fid);  
>> B = textscan(fid, '%f %f %f', 'CollectOutput', 1)  
B =  
    [2x3 double]  
>> fclose(fid);  
.....
```




数据写入TXT文件



MATLAB中写文本文件的常用函数

高级函数		低级函数	
函数名	说 明	函数名	说 明
save	将工作空间中的变量写入文件	fprintf	按指定格式把数据写入文件
dlmwrite	按指定格式将数据写入文件		



一、调用dlmwrite函数写入数据

调用格式:

dlmwrite(filename, M)

dlmwrite(filename, M, 'D')

dlmwrite(filename, M, 'D', R, C)

dlmwrite(filename, M, 'attrib1', value1, 'attrib2', value2, ...)


dlmwrite(filename, M, '-append')

dlmwrite(filename, M, '-append', attribute-value list)



dlmwrite函数支持的参数名与参数值列表

参数名	参数值	说 明
delimiter	单个字符, 如 ',', '"', '\t' 等	设定数据间分隔符
newline	'pc'	设定换行符为 '\r\n'
	'unix'	设定换行符为 '\n'
roffset	通常为非负整数	M 矩阵的左上角在目标文件中所处的行
coffset	通常为非负整数	M 矩阵的左上角在目标文件中所处的列
precision	以%号引导的精度控制符, 如 '%10.5f'	和 C 语言类似的精度控制符, 用来指定有效位数



【例6.2-1】 用逗号作为分隔符，调用dlmwrite函数将如下复数矩阵写入文件examp6_2_9.txt

```
>> x=[1.455390+1.360686i 8.692922+5.797046i 5.498602+1.449548i  
8.530311+6.220551i  
3.509524+5.132495i 4.018080+0.759667i 2.399162+1.233189i 1.839078+2.399525i  
4.172671+0.496544i 9.027161+9.447872i 4.908641+4.892526i  
3.377194+9.000538i];  
>> dlmwrite('examp6_2_9.txt', x, 'delimiter', ',', 'newline', 'pc')
```



二、调用fprintf函数写入数据

调用格式:

count = fprintf(fid, format, A, ...)


【例】

% 在屏幕上显示一句话

```
>> y = fprintf(1, '祝福我们伟大的新中国%d周岁生日快乐!!! ', 60)
```

```
祝福我们伟大的新中国60周岁生日快乐!!!
```

```
y =
```



【例6.2-2】用fprintf函数将数据写入文件examp6_2_01.txt至
examp6_2_11.txt的代码

```
>> x = 10*rand(8,5);
```

```
>> fid = fopen('examp6_2_01.txt','wt');
```

```
>> fprintf(fid,'%f %f %f %f %f %f %f %f\n', x);
```

```
>> fclose(fid); % 关闭文件
```

```
.....
```

调用fprintf函数写入数据或在屏幕上显示数据时，format参数指定的格式循环作用在矩阵的列上，原始矩阵的列在文件中或屏幕上就变成了行。