

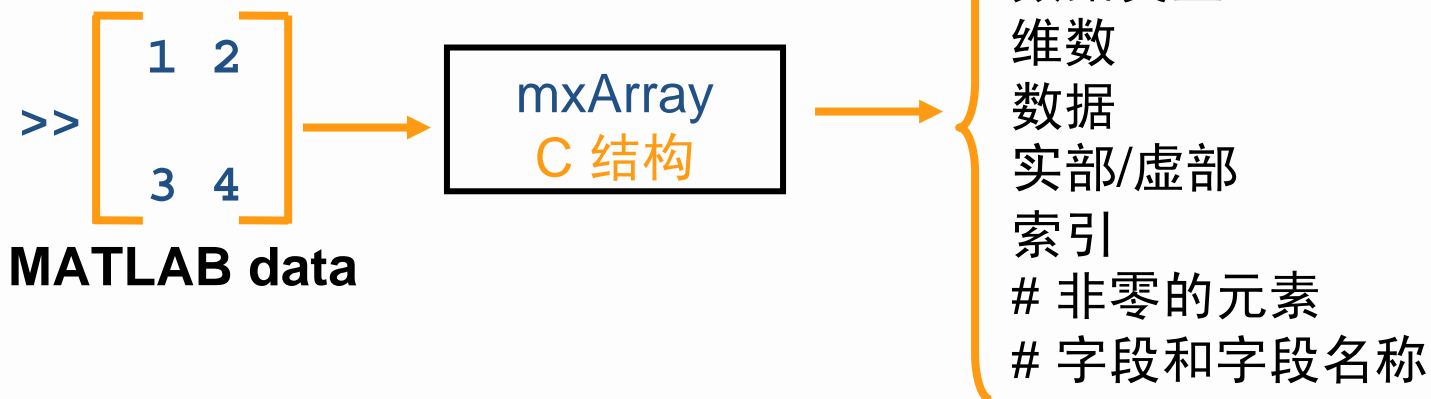
## MATLAB数据结构 mxArray



## 本章概述

- MATLAB 数据
- mxArray 数据类型
- mx 和 mex 前缀
- 数值与字符串数组 (创建/访问/查询)
- 通用函数(创建/设置/查询)
- 结构与元胞
- 参考：稀疏矩阵
- 参考：逻辑数组

## MATLAB 数据



## 数据的存储

► 列元素优先——与Fortran类似

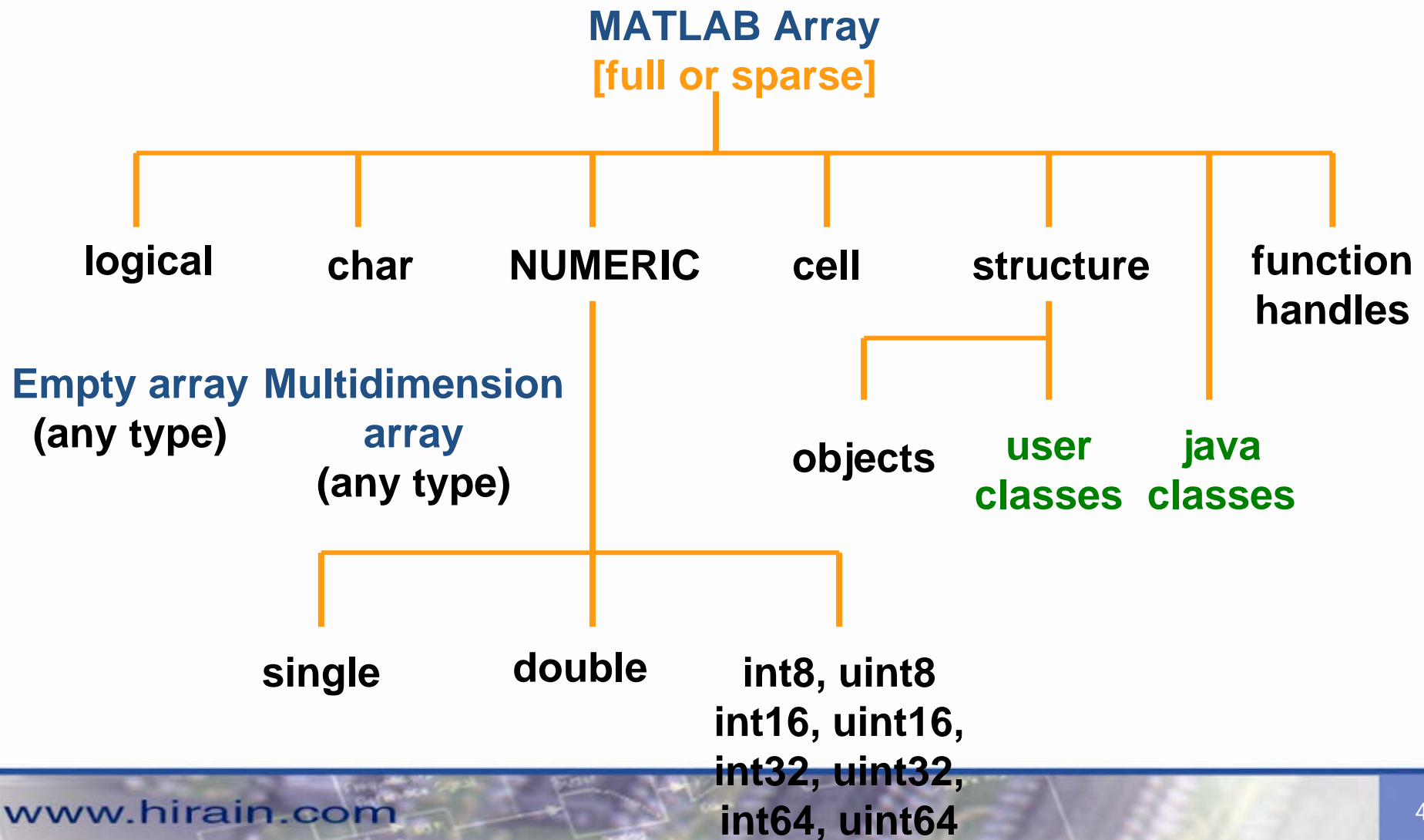
```
>> A=[1 2; 3 4];
```



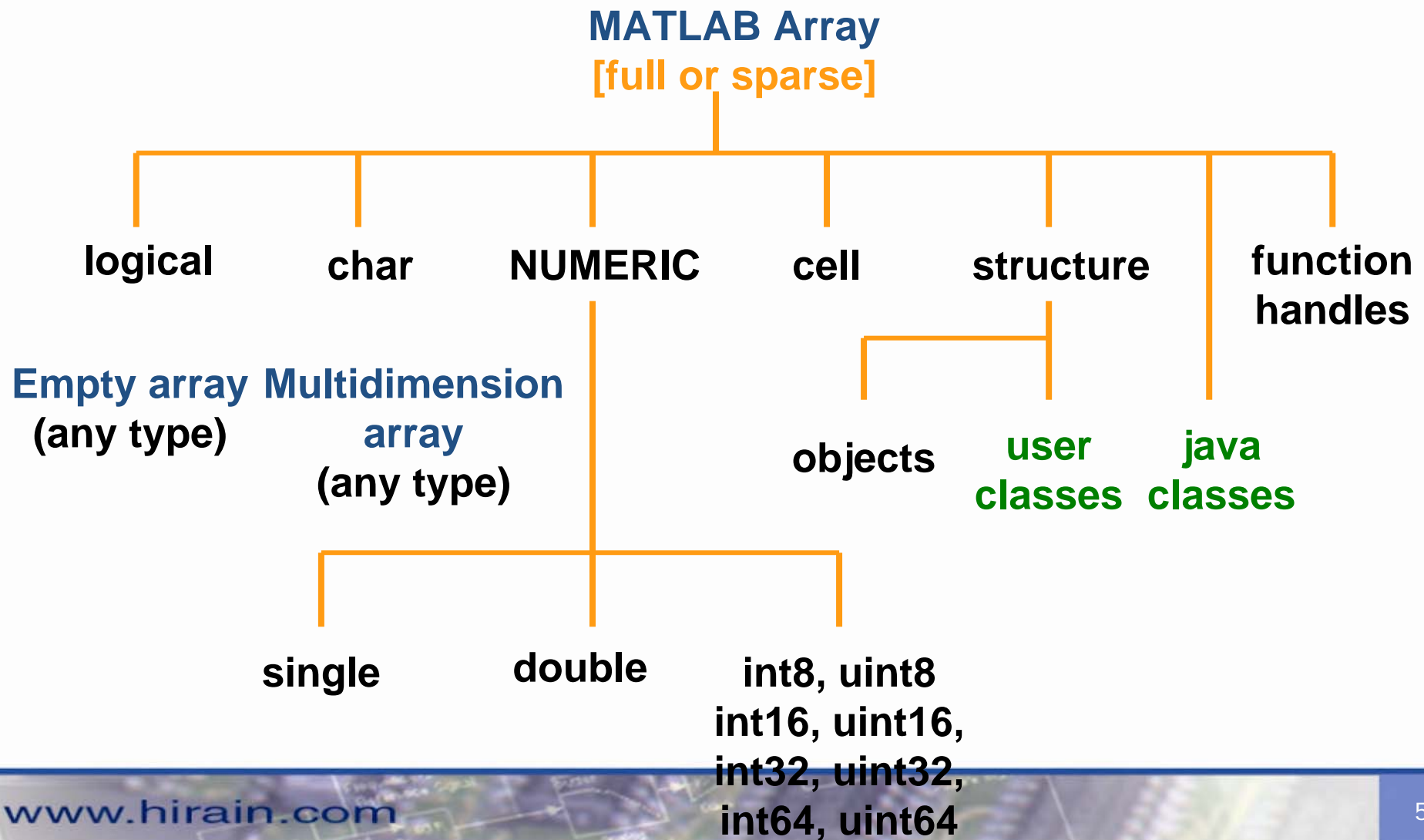
```
>> B=['Hello'; 'Users']
```



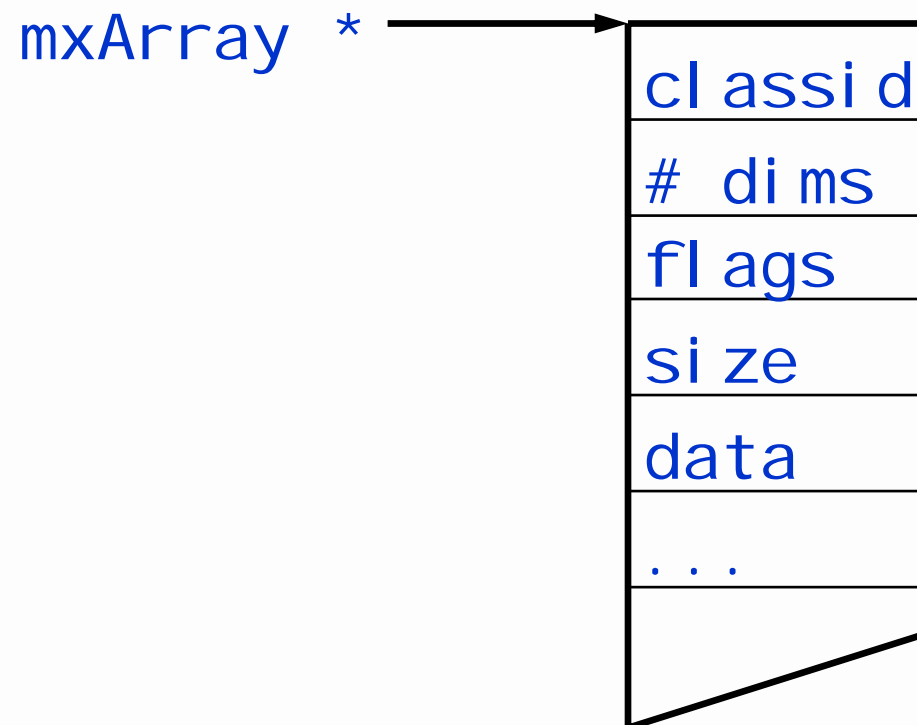
# MATLAB的数据类型



## MATLAB数据类型 (续)

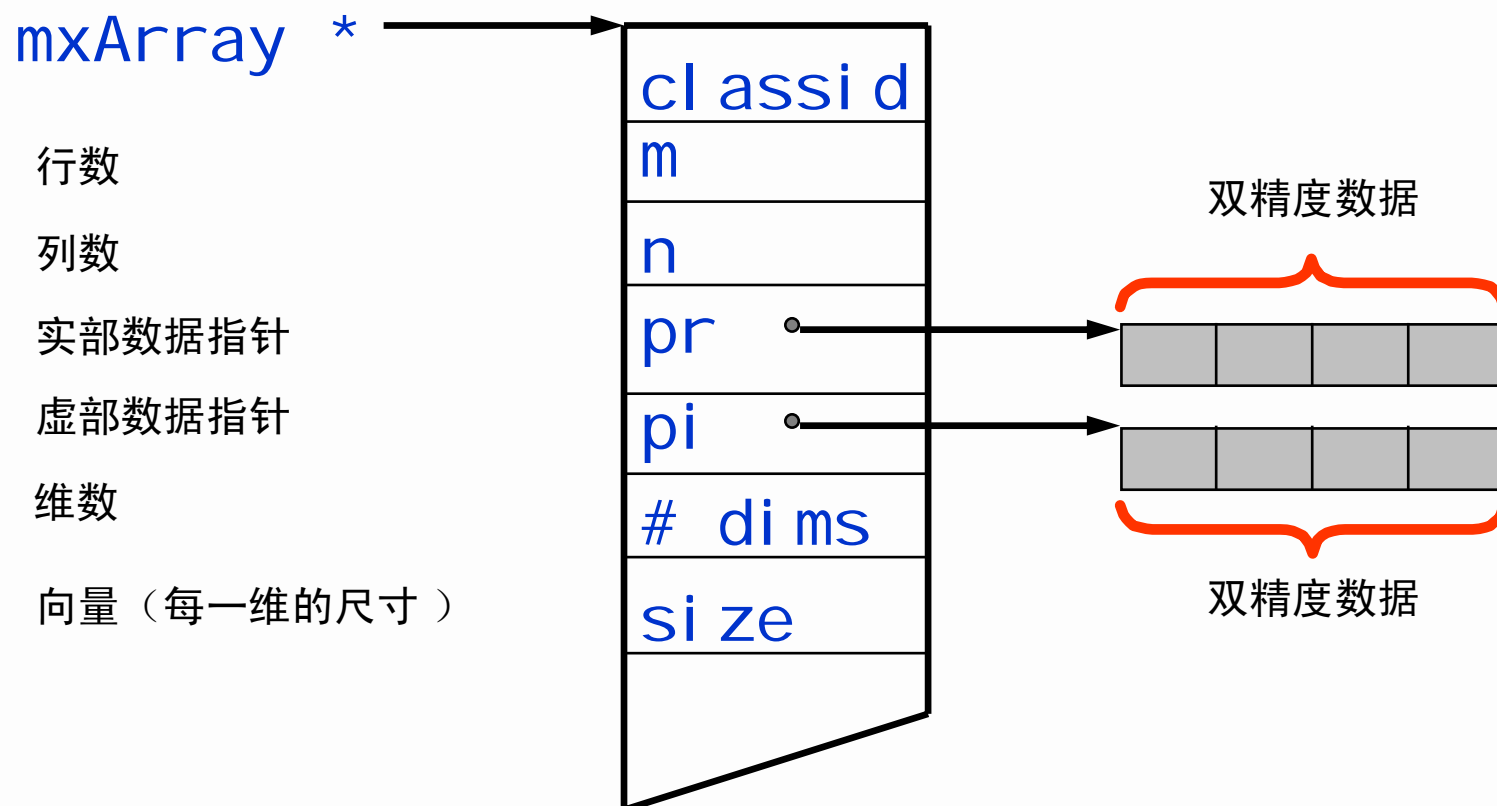


# mxArray



使用 get 和 set 辅助程序管理mxArray数据对象

## mxArray 类型 —— double



## 类型定义

```
typedef enum {  
    mxUNKNOWN_CLASS = 0,  
    mxCELL_CLASS,  
    mxSTRUCT_CLASS,  
    mxOBJECT_CLASS,  
    mxCHAR_CLASS,  
    mxLOGICAL_CLASS,  
    mxDOUBLE_CLASS,  
    mxSINGLE_CLASS,  
    mxINT8_CLASS,  
    mxUINT8_CLASS,  
    mxINT16_CLASS,  
    mxUINT16_CLASS,  
    mxINT32_CLASS,  
    mxUINT32_CLASS,  
    mxINT64_CLASS,  
    mxUINT64_CLASS,  
} mxClassID;
```

```
typedef enum {  
    mxREAL=0,  
    mxCOMPLEX  
} mxComplexity;
```

```
typedef uint16_T mxChar;
```

注意:

- 以mx为前缀的函数需要调用或者返回mxClassID枚举类型的变量



## 示例：了解MATLAB数据

源代码文件：

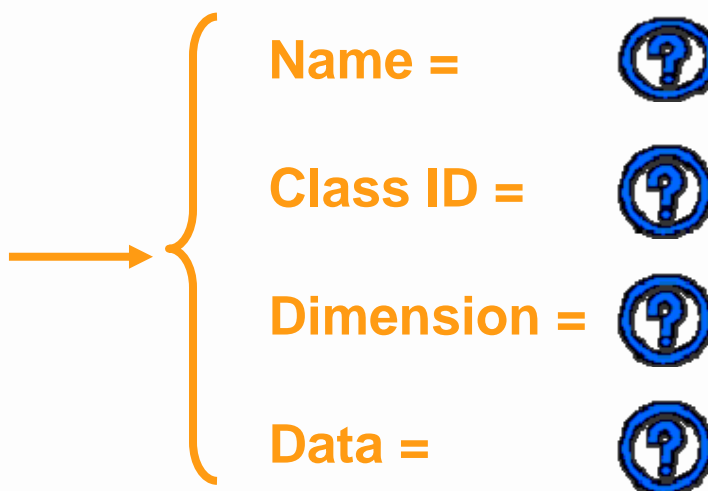
```
c:\class\coursefiles\ML02\explore.c
```

```
>> cd c:\class\coursefiles\ML02
```

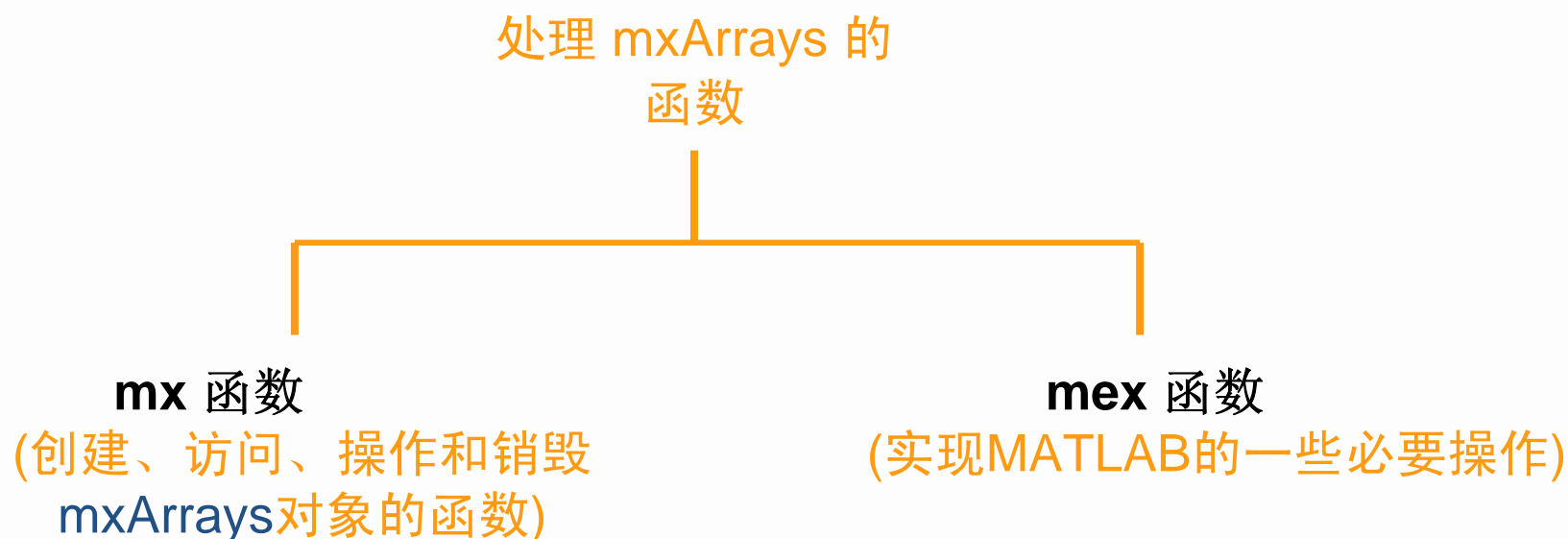
```
>> mex explore.c
```

```
>> x=2;
```

```
>> explore(x)
```



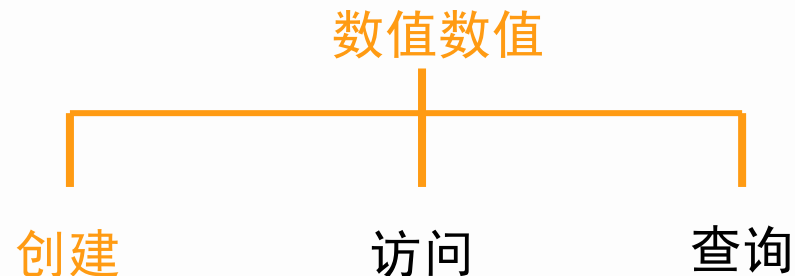
## mx 和 mex 前缀



注意:

- `mex` 仅仅在MEX文件中可用;
- `mex.h` 必须包含到C源文件;
- `mex.h` 包含了 `matrix.h` , 用于执行 `mx` 函数。

## 数值数组 – 创建



```
mxCreateDoubleMatrix( int, int, mxComplexity);
```

- 创建二维的 mxArray 双精度类型数据对象

```
mxCreateNumericMatrix(int, int, mxClassID, mxComplexity );
```

- 创建二维的mxArray数据对象，数据的类型可以被任意定义

```
mxCreateDoubleScalar( double );
```

- 创建双精度类型的mxArray标量数据对象
- 双精度数据作为输入参量，数据对象的初值为输入的参量数值

## 数值数组 – 创建(Continued)

```
mxCreateNumericArray( int,int *,mxClassID,mxComplexity);
```

- 创建N维的mxArray数据对象，数据类型可以任意定义

```
mxCreateSparse(int,int, int, mxComplexity);
```

- 创建双精度的二维稀疏矩阵mxArray数据对象
- 将ir, jc, nzmax 和附属标志作为输入参数

```
/* Code snippet from creatematrix.c */
```

```
...
```

```
void mexFunction(int nlhs,mxArray* plhs[],  
                  int nrhs,const mxArray* prhs[]) {  
    mxArray* mlInteger,* mlDouble;  
    mlInteger = mxCreateNumericMatrix(3,3,mxINT32_CLASS,mxREAL);  
    mlDouble = mxCreateDoubleMatrix(3,3,mxREAL);  
    ...  
    mxDestroyArray(mlDouble);  
}
```

## 数值数组 – 访问



`mxGetPr ( const mxArray * ) & mxGetPi ( const mxArray * )`

- 获取实部和虚部数据的指针

`mxGetData ( const mxArray * )`

- 获取mxArray数据对象的数据，数据类型为非双精度类型

`mxGetImagData ( const mxArray * )`

- 获取mxArray数据对象的虚部数据，数据类型为非双精度类型

`mxGetScalar ( const mxArray * )`

- 获取mxArray数据对象的第一个元素的指针，数据类型为非元胞或结构数据

## 数值数组 – 访问(续)

```
mxGetIr ( const mxArray * ), mxGetJc ( const mxArray * ) &  
mxGetNzmax(const mxArray * )
```

■ 获取mxArray稀疏矩阵对象的相关参数

```
/* Code snippet from creatematrix.c */
```

```
...
```

```
void mexFunction(int nlhs,mxArray* plhs[],  
                 int nrhs,const mxArray* prhs[])  
{  
    ...  
    memcpy(mxGetData(mInteger),cInt,sizeof(cInt));  
    memcpy(mxGetPr(mDouble),cDbl,sizeof(cDbl));  
    ...  
}
```

## 数值数组 – 查询类型



- 所有数值类型数组都可以通过一系列mxIs开头的函数查询其类型
- 所有mxIs函数返回值均为Boolean类型变量

- 常用的函数包括:

mxIsComplex(const mxArray \*)

mxIsDouble(const mxArray \*)

mxIsLogical(const mxArray \*)

mxIsInt8(const mxArray \*)


## 数值数组 – 查询类型(续)

```
/* Code snippet from querymatrix.c */  
  
...  
  
void mexFunction(int nlhs, mxArray * plhs[],  
                 int nrhs, const mxArray * prhs[])  
{  
    ...  
    mxComplexity cmplx;  
  
    /* Get the complexity of the mxArray */  
    cmplx = (mxIsComplex(prhs[0])  
            ? mxCOMPLEX : mxREAL);  
    ...  
}
```



## 示例：创建数值数组

```
/* File creatematrix.c */
#include "mex.h"
void mexFunction(int nlhs,mxArray* plhs[],int
    nrhs,const mxArray* prhs[])
{
    mxArray* mInteger,* mDouble;
    /* Declare C data */
    ...
    /* Check errors for inputs and outputs */
    ...
    /* Create the numeric mxArray */
    ...
    mInteger = mxCreateNumericMatrix(3,3,
                                    mxINT32_CLASS,mxREAL);
    mDouble = mxCreateDoubleMatrix(3,3,mxREAL);
}
```



## 示例：创建数值数组(续)

```
/* Copy data from the C arrays mxArray */  
memcpy(mxGetData(mInteger),cInt,sizeof(cInt));  
memcpy(mxGetPr(mDouble),cDbl,sizeof(cDbl));  
/* Return the mxArray as output arguments */  
plhs[0] = mInteger; /* Return 1st output */  
if (nlhs == 2)  
    plhs[1] = mDouble;  
else    mxDestroyArray(mDouble); /* Recommended */  
}
```

```
>> mex creatematrix.c  
>> creatematrix  
>> [A,B] = creatematrix
```

## 字符数组 – 创建



`mxCreateString( const char * );`

- 将C语言字符串作为输入初始化mxArray数据对象

`mxCreateCharArray( ndims, const int * );`

- 将矩阵的维数和每一维的长度作为输入参数，后者为整数类型的数组

`mxCreateCharMatrixFromStrings( rows, const char ** );`

- 将C语言字符串数组和相应的行列数作为输入参数

## 字符数组 – 创建 (续)

```
/* Code snippet from createstring.c */
```

```
...
```

```
void mexFunction(int nlhs,mxArray* plhs[],  
                 int nrhs,const mxArray* prhs[])
```

```
{
```

```
    /* Declare and define your C style strings */
```

```
    ...
```

```
    mlString1 = mxCreateCharMatrixFromStrings(5,cStr);
```

```
    mlString2 = mxCreateString(cSimpleStr);
```

```
    ...
```

```
}
```

## 字符数组 – 访问



`mxGetChars( const mxArray * );`

- 将字符类型的mxArray 数据对象作为输入参数
- 返回mxArray数据对象第一个字符的地址

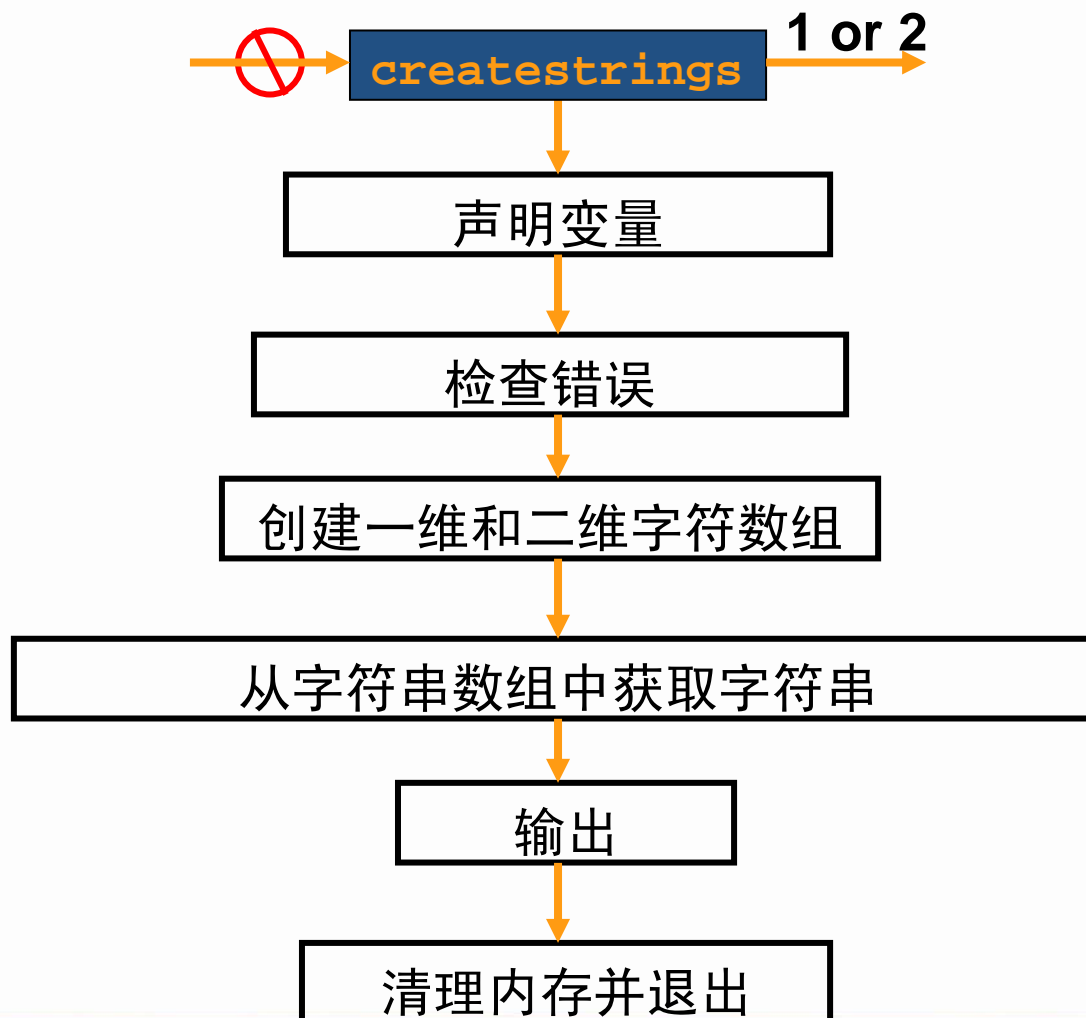
`mxGetString( const mxArray *, char *, int );`

- 将字符类型的mxArray数据对象、字符串的指针和字符串的长度作为输入参数，将数据对象的字符串拷贝到相应的指针中
- 返回值为0则函数运行成功，否则返回1

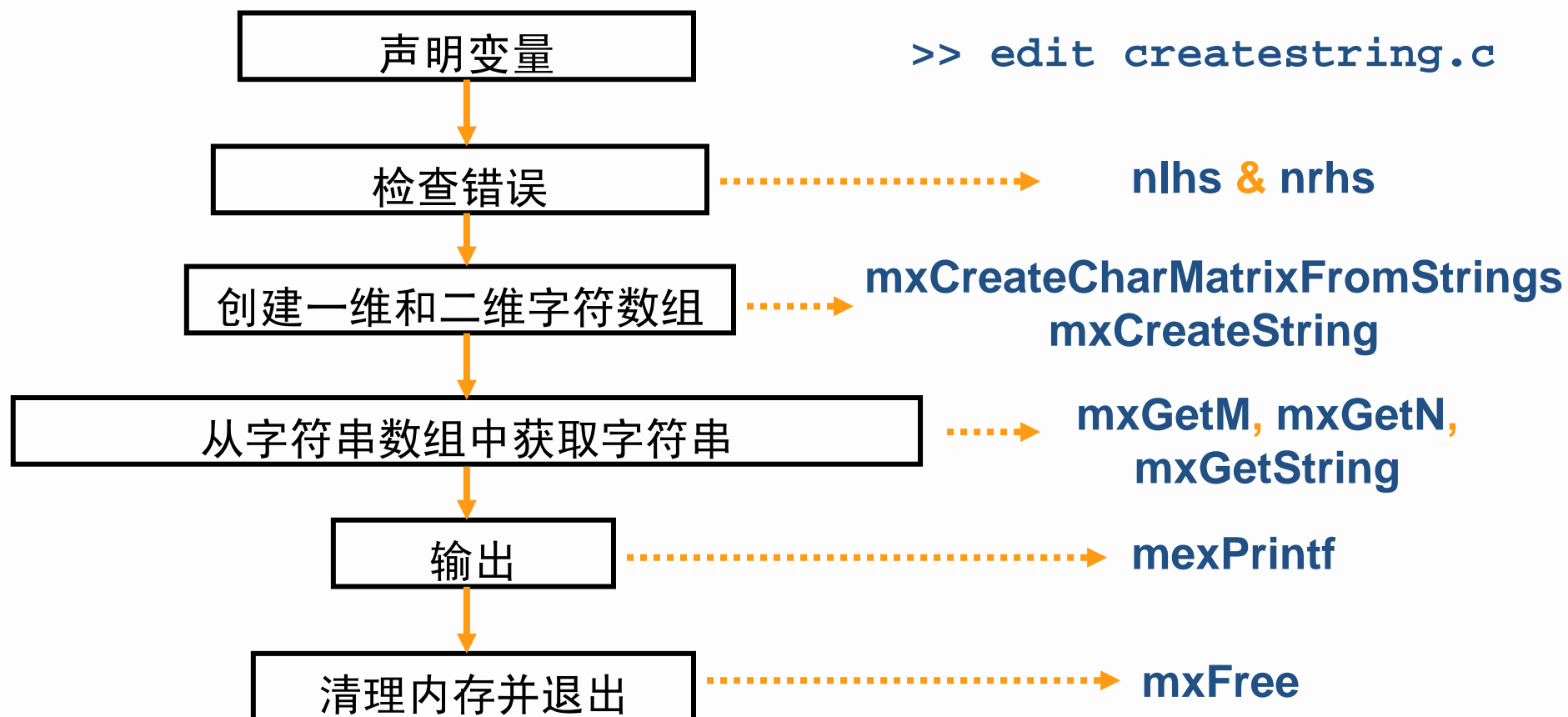
## 字符数组 – 访问 (续)

```
/* Code snippet from createstring.c */
...
void mexFunction(int nlhs,mxArray* plhs[],
                 int nrhs,const mxArray* prhs[])
{
    /* Declare & define your C style strings & buffer*/
    /* Create mxArray */
    ...
    buflen=(mxGetM(mlString2)*mxGetN(mlString2)+ 1);
    buffer = mxMalloc(buflen, sizeof(mxChar));
    flag = mxGetString(mlString2,buffer,buflen);
    ...
    mxFree(buffer);
}
```

## 示例：创建并访问字符串数组



## 示例：创建并访问字符串



```
>> edit createstring.c
```

`nlhs & nrhs`

`mxCreateCharMatrixFromStrings`  
`mxCreateString`

`mxGetM`, `mxGetN`,  
`mxGetString`

`mexPrintf`

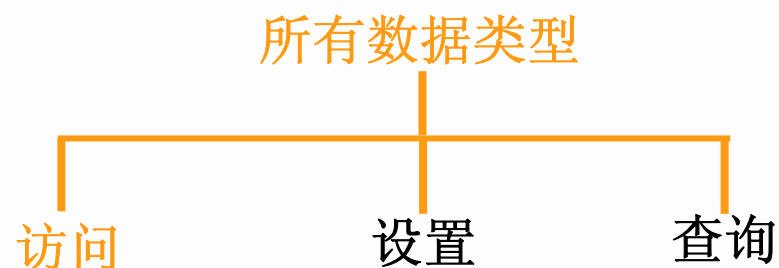
`mxFree`



Exercise: Modify Matrices



## 通用函数 – 访问数据



- 所有通用的访问数据函数都以 `mxGet` 开头，后面紧随着感兴趣的属性或者字段
- 大多数 `mxGet` 函数的输入参数都是 `const mxArray *`

- 常用的函数包括：

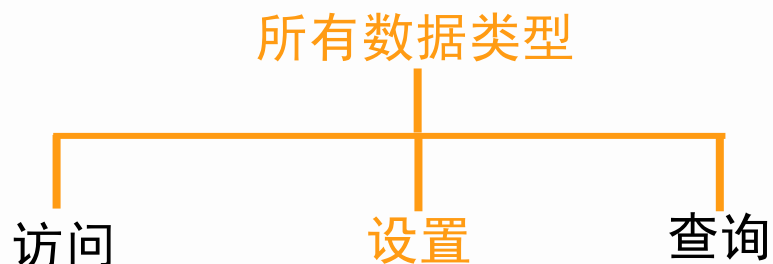
`mxGetClassName(const mxArray *)`

`mxGetClassID(const mxArray *)`

`mxGetNumberOfElements(const mxArray *)`

`mxGetM(const mxArray *) / mxGetN(const mxArray *)`

## 通用函数 – 设置数据



- 所有设置数据的函数都以mxSet开头，后面是相应的属性或字段
- 大多数 mxSet函数的输入参数是 mxArray \*

- 常用的函数包括：

`mxSetData(mxArray *, void *)`

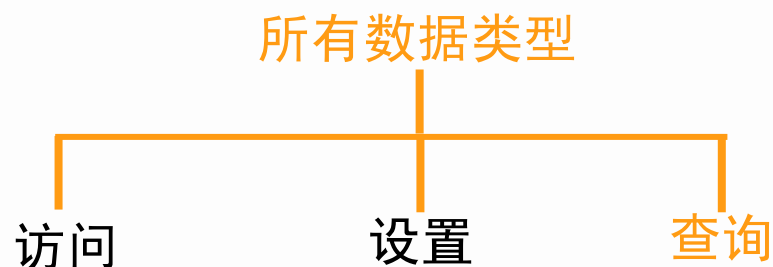
`mxSetM(mxArray *, int)`

`mxSetN(mxArray *, int)`

`mxSetPr(mxArray *, double *)`

`mxSetPi(mxArray *, double *)`

## 通用函数 – 查询数据



- 所有查询数据的函数都以 `mxIs` 开头
- 所有 `mxIs` 函数的返回值为 `Boolean` 类型的变量

- 常用的函数包括

```
mxIsClass(const mxArray *, const char *)  
mxIsCell(const mxArray *)  
mxIsChar(const mxArray *)  
mxIsStruct(const mxArray *)  
mxIsFromGlobalWS(const mxArray *)
```

## 示例：矩阵转置

本例子是用前面介绍的访问和查询函数实现下列功能：

- 识别输入参数的类型
- 对输入参数进行转置处理。
- 输出转置结果。

- 需要使用下列mx函数：

`mxGetClassID`

`mxIsComplex`

`mxGetM` / `mxGetN`

`mxGetClassName`

`mxCreateNumericMatrix`

`mxGetPr` / `mxGetPi`

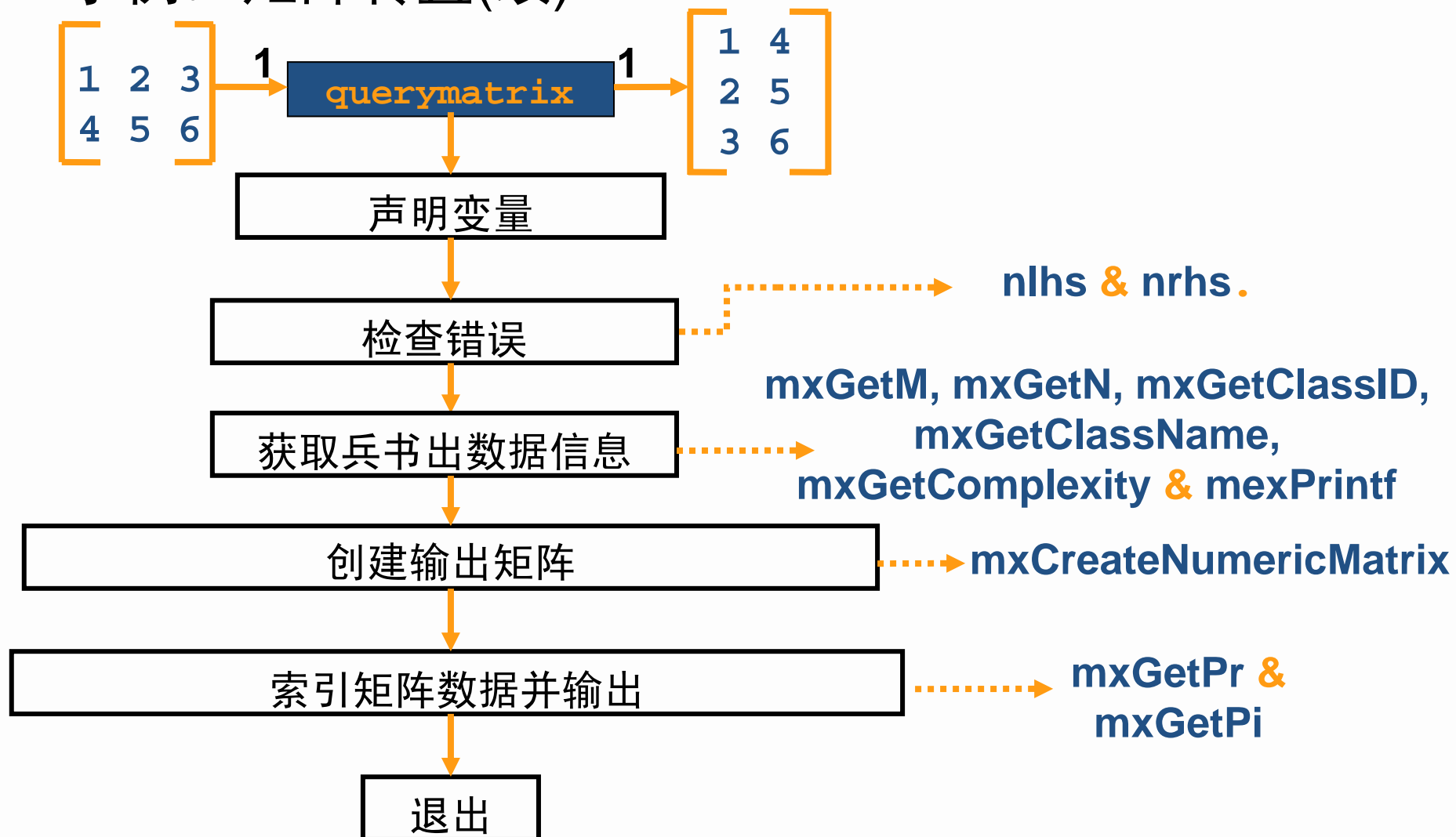
$$>> \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
>> edit querymatrix.c
```

```
>> mex querymatrix.c
```

```
>> querymatrix([1 2 3;...  
                4 5 6;])
```

## 示例：矩阵转置(续)



## 结构 – 创建



**`mxCreateStructArray( int, int *, int, const char **)`**

- 创建mxArray结构空数据对象
- 返回mxArray数据对象指针

**`mxCreateStructMatrix( int, int, int, const char **)`**

- 创建二维结构mxArray数据对象
- 返回 mxArray数据对象

## 结构 – 访问数据



**mxGetField( const mxArray \*, int, const char \*)**

- 获取指定的结构中指定字段的数值

**mxGetFieldName( const mxArray \*, const char \*)**

- 获取指定结构中的指定字段名称的字段序号
- 如果返回值为-1则字段不存在

**mxGetFieldByNumber( const mxArray \*, int, int)**

- 获取指定结构中指定索引字段的数值，输入参数还需要字段的数量

**mxGetFieldNameByNumber( const mxArray \*, int)**

- 获取指定结构中的指定序号的字段名称

**mxGetNumberOfFields (const mxArray \*)**

- 获取字段的数量

## 结构 – 设置数据



`mxSetField( mxArray *,int,const char *, mxArray *)`

- 根据给定的mxArray对象、索引和字段名称，设置字段的数值

`mxSetFieldByNumber( mxArray *,int,int,mxArray *)`

- 根据给定的mxArray数据对象、字段数量和索引设置字段的数值

`mxAddField( mxArray *,const char *)`

- 按照给定的字段名称给mxArray数据对象添加字段

`mxRemoveField( mxArray *, const char *)`

- 从mxArray对象中删除一个字段



## 示例：创建结构

本例子将利用前面介绍的函数实现：

- 创建结构类型的数据，该结构具有两个字段
- 设置记录数据
- 获取字段的序号等信息
- 将创建 mxArray数据对象返回



- 可能需要用到的函数：

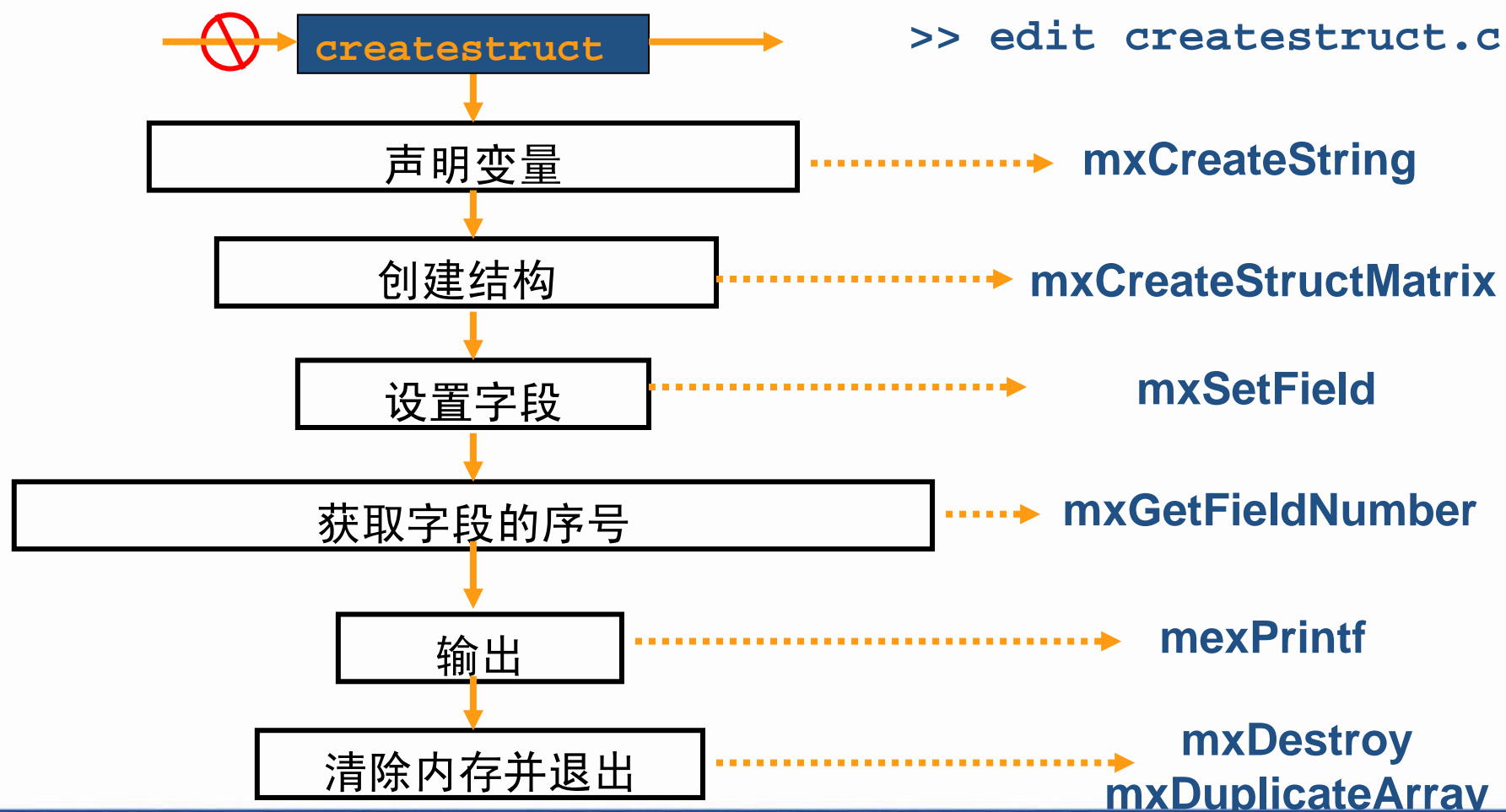
```
mxCreateStructMatrix  
mxCreateString  
mxSetField  
mxGetFieldNumber  
mexPrintf
```

Structure

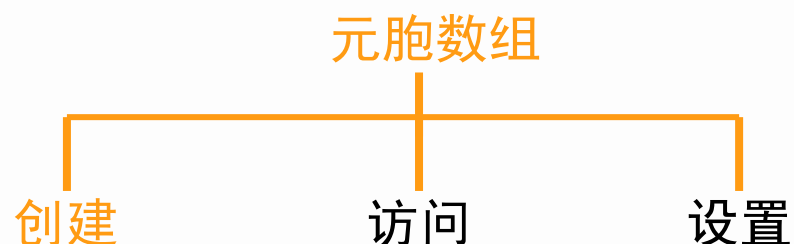
```
└─ FileName1 (FieldValue1)  
   FileName2 (FieldValue2)
```

```
>> edit createstruct.c  
>> mex createstruct.c  
>> which createstruct  
>> createstruct
```

## 示例：创建结构 (续)



## 元胞数组 – 创建



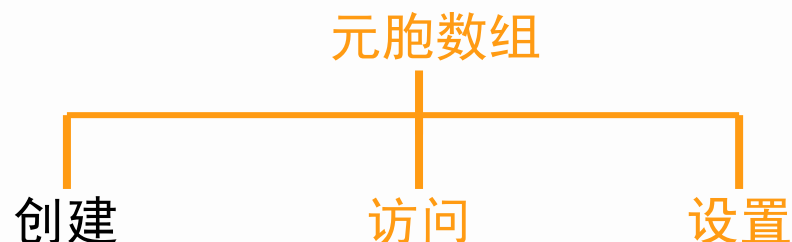
**`mxCreateCellArray( int, int * )`**

- 创建一个空元胞数组mxArray对象
- 返回mxArray数据对象指针

**`mxCreateCellMatrix( int, int )`**

- 创建二维元胞数组mxArray对象
- 返回mxArray数据对象的指针

## 元胞数组 – 访问和设置



### 访问

`mxGetCell( const mxArray *, int )`

- 根据索引获取mxArray元胞数组的元素

### 设置

`mxSetCell( const mxArray *, int, mxArray *)`

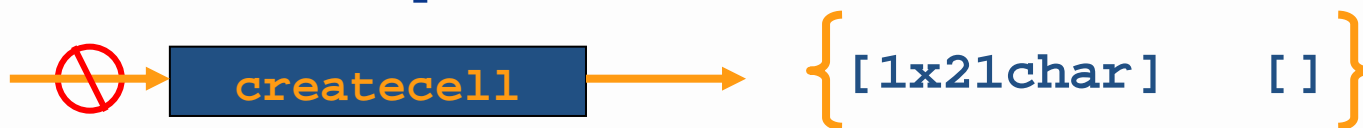
- 设置元胞数组元素，元素的内容是mxArray数据对象

注意：元胞数组的第一个元素索引是0

## 示例：创建元胞数组

本例子将利用前面介绍的函数创建设置访问元胞类型的数组：

- 创建尺寸为1x2的元胞数组
- 设置第一个元胞为字符串
- 获取字符串信息并输出
- 返回创建mxArray数据对象

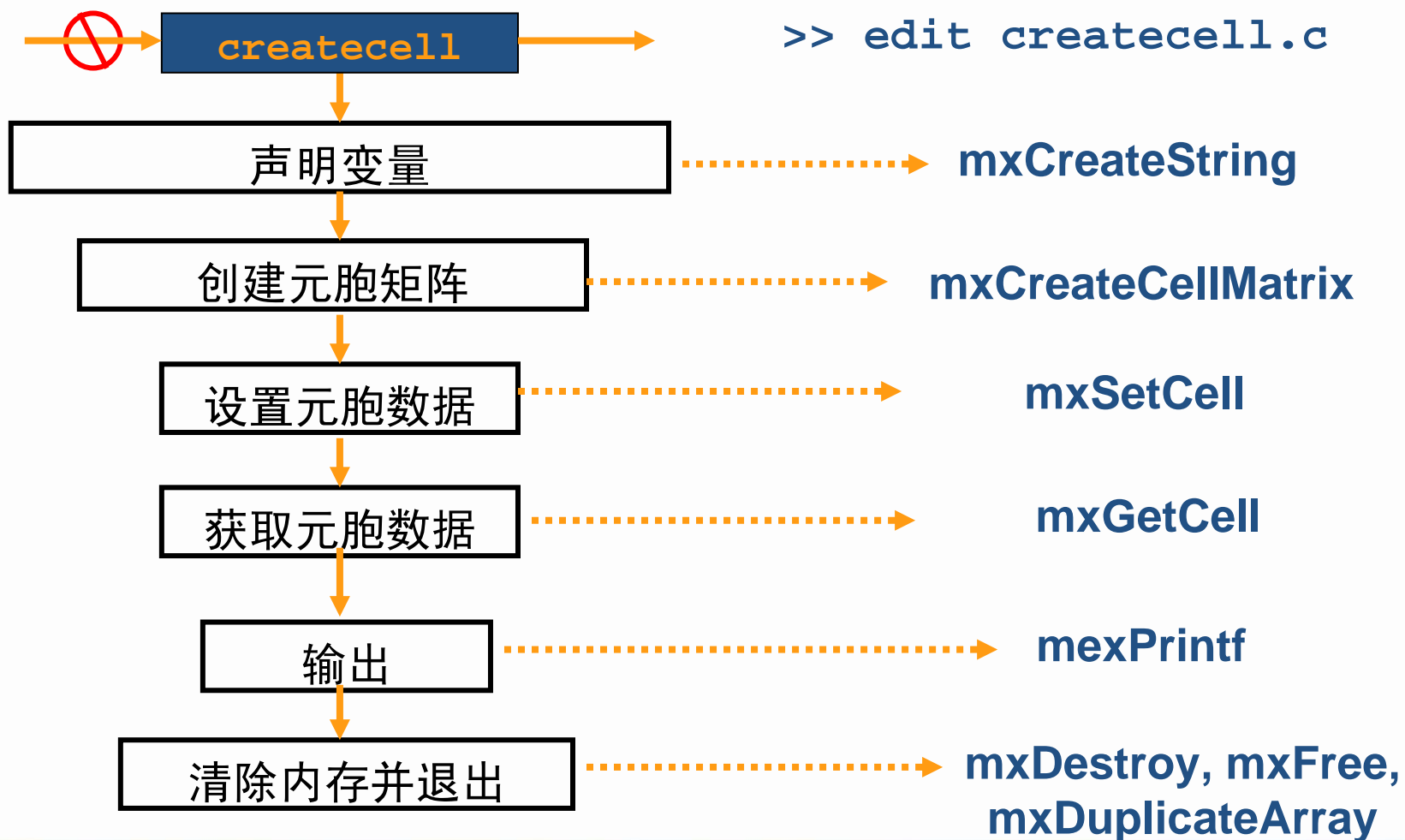


- 本例子中可能会用得的函数：

```
mxCreateCellMatrix  
mxSetCell  
mxGetCell  
mxCreateString  
mxGetString  
mexPrintf
```

```
>> edit createcell.c  
>> mex createcell.c  
>> which createcell  
>> createcell
```

## 示例：创建元胞数组(续)



## 本章小结

- MATLAB 数据
- mxArray 数据类型
- mx 和 mex 前缀
- 数值与字符串数组 (创建/访问/查询)
- 通用函数(创建/设置/查询)
- 结构与元胞
- 参考：稀疏矩阵
- 参考：逻辑数组

## 参考：稀疏矩阵

$$\begin{pmatrix} 3 & 0 & 10 & 0 \\ 0 & 0 & -7 & 0 \\ 0 & 0 & 2 & 0 \\ 1 & 0 & 0 & -2 \end{pmatrix}$$

满阵要占据  
 **$25 \times 8 = 200$  bytes**

稀疏矩阵占用  
**84 bytes**

*ir* 整数类型数组，长度为 *nnz*

► 包含 *pr* 数组中元素的行索引值

•  $ir \Rightarrow 0, 3, 0, 1, 2, 3$

*jc* 整数类型数组，长度为  $Cols + 1$

► 包含 *ir* 数组中第一个非零元素的索引和该列元素个数的和

•  $jc \Rightarrow 0, 2, 2, 5, 6$



## 参考：稀疏矩阵 – 创建



`mxCreateSparse(int, int, int)`

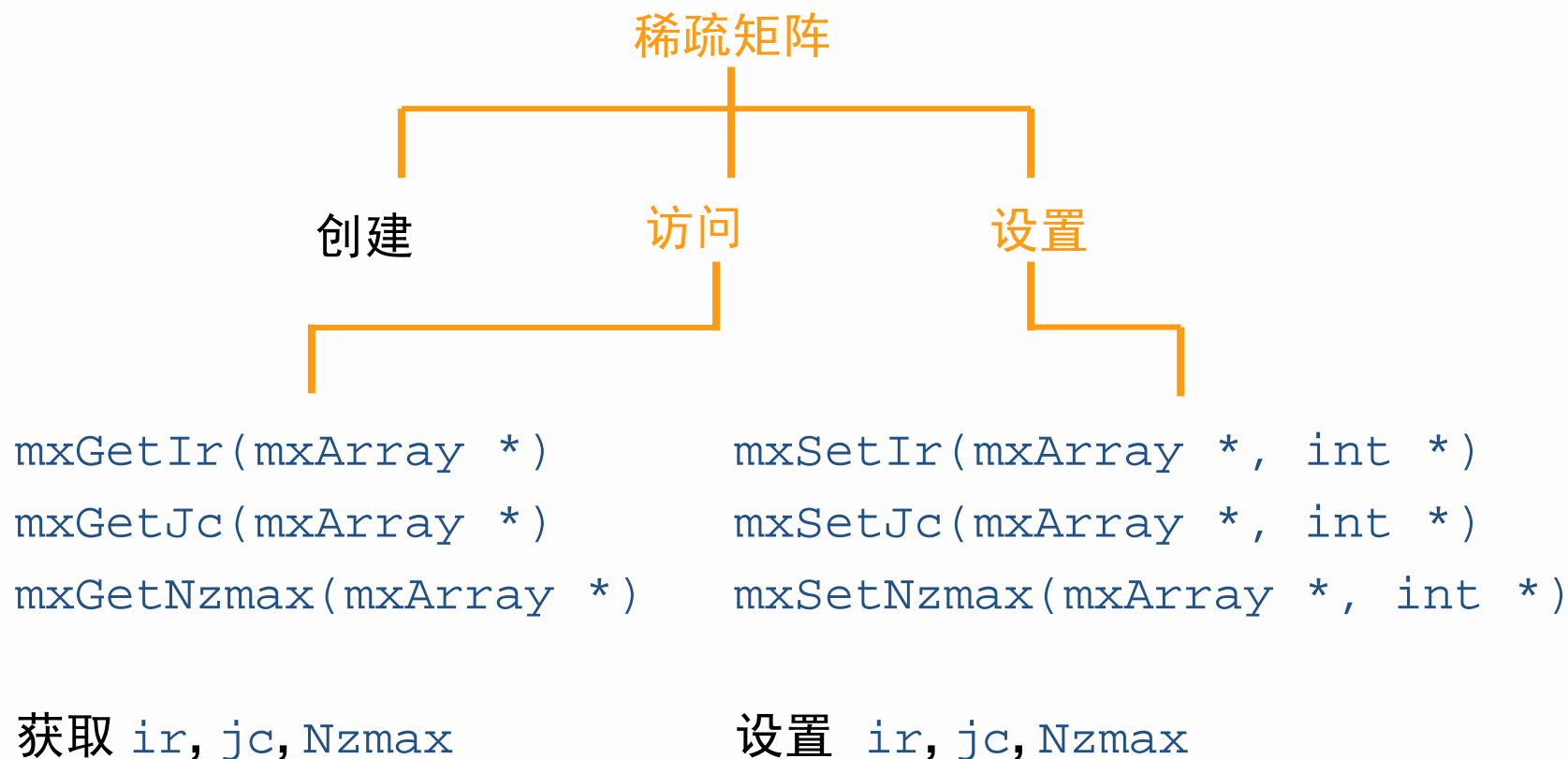
- 创建二位稀疏矩阵

`mxCreateSparseLogicalMatrix(int, int, int)`

- 创建二维逻辑类型稀疏矩阵

- 上述两个函数都需要ir,jc,nzmax等参数创建矩阵

## 参考：稀疏矩阵 – 访问和设置



## 示例：创建稀疏矩阵

本例子利用前面介绍的函数创建稀疏矩阵

- 创建稀疏矩阵
- 设置 `ir`, `jc`和 `pr`.
- 返回创建的结果.

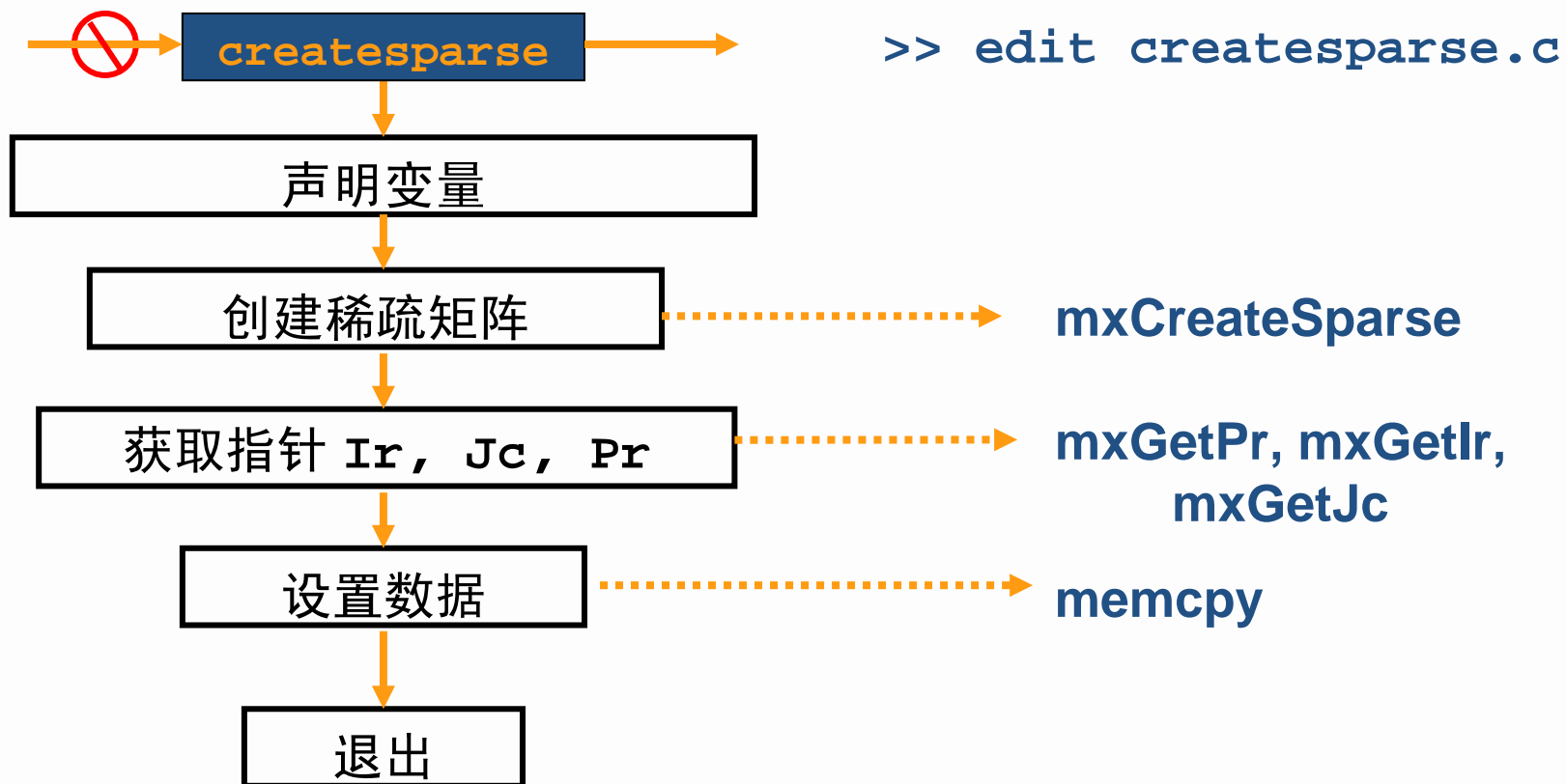


- 可能用到的函数有：

```
mxCreateSparse  
mxGetPr  
mxGetIr  
mxGetJc  
memcpy
```

```
>> edit createsparse.c  
>> mex createsparse.c  
>> which createcell  
>> createsparse
```

## 示例：创建稀疏矩阵(续)



## 参考：逻辑数组

**`mxArray *mxCreateLogicalScalar(mxLOGICAL);`**

- 该函数输入参数的类型为mxLOGICAL，这种数据类型就是C语言中表示布尔类型的数据类型，一般为bool。该函数的输出参数是mxArray数据类型的对象，它表示逻辑量。

**`mxArray *mxCreateLogicalMatrix(int , int );`**

- 该函数的输入参数是逻辑矩阵的行数m和列数n，函数的输出参数是mxArray数据类型的对象，表示逻辑量矩阵。

**`mxArray *mxCreateLogicalArray(int, const int *);`**

- 该函数的输入参数是数组的维数ndim和每一维的尺寸ndims，函数的输出参数是mxArray数据类型的对象，表示逻辑数组。
- 例如：

```
/* 创建mxArray数据对象-> 逻辑类型数组 */  
Data = mxCreateLogicalArray(2,ndims);  
/*获取数据的指针*/  
pr = mxGetLogicals(Data);  
/* 通过内存赋值的方法完成数据的赋值 */  
memcpy(pr,data,12*sizeof(mxLogical));
```

