CELL ARRAY

Special MATLAB array whose elements are containers that can hold other MATLAB arrays

- Different kinds of information can be kept together and accessed by a single name
- Brace ‘{ }’ is used for selecting and displaying the contents

```matlab
>> a(1, 1)
an ans = [3x3 double]
>> a{1, 1}
an ans =
1 3 -7
2 0 6
0 5 1
```

Data structure: 

```
[ 1 3 -7]
[ 2 0 6]
[ 0 5 1]
```

Content: 

```
[ ]
```

```
[3 + 4i  -5 ]
[ -10i 3 -4i]
```
Content indexing

\[
a\{1, 1\} = \begin{bmatrix} 1 & 3 & -7; 2 & 0 & 6; 0 & 5 & 1 \end{bmatrix};
a\{1, 2\} = \text{‘This is a text string.’};
a\{2, 1\} = \begin{bmatrix} 3+4i & -5; -10i & 3 & -4i \end{bmatrix};
a\{2, 2\} = \emptyset;
\]

Define the contents of the data structure contained in a cell

Create a data structure containing the specified data and then assigns that data structure to a cell
CREATING CELL ARRAYS

Create empty cell

```matlab
a = cell(2, 2)
```

Using a cell constructor

```matlab
b = {[1 2], 17, [2; 4]; 3-4*i, 'Hello', -3}
```
VIEWING THE CONTENTS OF CELL ARRAYS

MATLAB displays the **data structure** not data itself

- Entire data structure, if it can be displayed in a single line
- Summary, otherwise

```matlab
>> a
a =
    [3x3 double]     [1x22 char]
    [2x2 double]     [ ]
>> b
B =
    [1x2 double]     [ 17]     [2x1 double]
    [3.0000- 4.0000i] 'Hello'     [ -3]
```

```matlab
>> celldisp(a)
a{1, 1} =
    1   3   -7
    2   0   6
    0   5   1
a{2, 1} =
    ...  
a{1, 2} =
    'This is a text string.'
a{2, 2} =
    [ ]
```
EXTENDING CELL ARRAYS

\[ \alpha(3, 3) = 5 \]

<table>
<thead>
<tr>
<th>cell 1,1</th>
<th>cell 1,2</th>
<th>cell 1,3</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
1 & 3 & -7 \\
2 & 0 & 6 \\
0 & 5 & 1
\end{bmatrix}
\] | 'This is a text string.' | |

<table>
<thead>
<tr>
<th>cell 2,1</th>
<th>cell 2,2</th>
<th>cell 2,3</th>
</tr>
</thead>
</table>
| \[
\begin{bmatrix}
3+i4 & -5 \\
-i10 & 3-i4
\end{bmatrix}
\] | | |

<table>
<thead>
<tr>
<th>cell 3,1</th>
<th>cell 3,2</th>
<th>cell 3,3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>[5]</td>
</tr>
</tbody>
</table>
DELETING CELLS IN ARRAYS

Assigning an empty cell

```matlab
>> a
a =
    [3x3 double]    [1x22 char]    []
    [2x2 double]    []    []
    []    []    [5]
>> a(3, :) = []
a =
    [3x3 double]    [1x22 char]    []
    [2x2 double]    []    []
    []    []    [5]
```
**USING DATA IN CELL ARRAYS**

C = \{[1 2; 3 4], ‘dogs’; ‘cats’, 1\}

>> c{1, 1}
ans =
   1   2
   3   4

>> a{2, 1}
ans =
cats

>> c{1, 1}(1, 2)
ans =
   2
**STRUCTURE ARRAYS**

**Structure** is a data type in which each individual element has a name

- Each element is called as **field**
- Each field may have a different type
- Structure is used to represent multiple properties of a single entity
  
  Eg.) Student has such properties as name, id, major, address, and etc.
  
  → student.name

**Structure array**

- An array of structures
- Each structure in the array has identically the same fields
  
  → student(1).name, student(2).id
CREATING STRUCTURE ARRAYS

A field at a time

>> student.name = 'Albert';
>> student.major = 'Computer Science';
>> student.id = 201500000

student =
    name : 'Albert'
    major : 'Computer Science'
    id : 201500000

>> student(2).name = 'Jane'

student =
1x2 struct array with fields:
    name
    major
    id

>> student(1)
ans =
    name : 'Albert'
    major : 'Computer Science'
    id : 201500000

>> student(2)
ans =
    name : 'Jane'
    major : []
    id : []
Creating structures with the `struct` function

- `str_array = struct('field1', val1, 'field2', val2, ...)`

```matlab
>> student(2) = struct('name', [], 'major', [], 'id', [])
student =
1x2 struct array with fields:
    name
    major
    id
```
ADDING FIELDS TO STRUCTURES

```matlab
>> student(2).exams = [95 100 100]
student =
1x2 struct array with field:
    name
    major
    id
    exams
```

```matlab
 >> student(1)
ans =
    name : ‘Albert’
    major : ‘Computer Science’
    id    : 201500000
    exams : [ ]
```

```matlab
 >> student(2)
ans =
    name : ‘Jane’
    major : [ ]
    id    : [ ]
    exams : [95 100 100]
```
>> student = rmfield(student, 'major')
student =
1x2 struct array with field:
    name
    id
    exams
USING DATA IN STRUCTURE ARRAYS

>> student(2).major
ans =
Computer Science
>> student(2).exams
ans =
   [95 100 100]
>> student(2).exams(1)
ans =
   95
>> mean(student(2).exams)
ans =
   98.3333

>>[student.name]
Ans =
   Albert    Jane
**GETFIELD/SETFIELD**

\[
f = \text{getfield}(\text{array}, \{\text{array\_index}\}, \text{‘field’}, \{\text{field\_index}\})
\]

\[
f = \text{array}(\text{array\_index}).\text{field}(\text{field\_index});
\]

\[
\text{setfield}(\text{array}, \{\text{array\_index}\}, \text{‘field’}, \text{value})
\]

\[
\text{array}(\text{array\_index}).\text{field}(\text{field.index}) = \text{value};
\]

\[
>>> \text{id} = \text{getfield}(\text{student}, \{1\}, \text{‘id’})
\]

\[
\text{Id} =
\begin{align*}
&201500000
\end{align*}
\]
DYNAMIC FIELD NAME

>> student(1).major
ans =
Computer Science
>> student(1).('major')
an =
Computer Science

funcion ave = calc_average(struct, field)
arr = [];
for ii = 1:length(structure)
    arr = [arr structure(ii).(field)]
end
ave = mean(arr);

>> ave = calc_average(student, 'exams')
Ave =
    98.3333
>> ave = calc_average(student, 'id')
Ave =
    201500000
NESTING STRUCTURE ARRAYS

>> student(1).class(1).name = ‘ECP 2015’;
>> student(1).class(2).name = ‘Math’;
>> student(1).class(1).instructor = ‘John’;
>> student(1).class(2).instructor = ‘Brown’

>> student(1)
ans =
    name : ‘Albert’
    major : ‘Computer Science’
    id : 201500000
    exams : [ ]
    class : [1x2 struct]

>> student(1).class
ans =
1x2 struct array with fields:
    name
    instructor
>> student(1).class(1)
ans =
    name : ‘ECP 2015’
    instructor : ‘John’
>> student(1).class(2).instructor
ans =
    Brown