Advanced capabilities of MATLAB

By

Dr. R. R. Manza

Objectives

- Introduction
- Cell Array
- Structure
- Sparse Array
- Summary

Introduction

This chapter introduces the advanced capabilities of MATLAB like cell array, structure, sparse array. Cell Array is the container, which provides a hierarchical storage mechanism for the different types of data. You can store the data of different types or different dimensions or size within a cell of the cell array. Different ways to create and handle the cell array are explained in this chapter.

Cell array

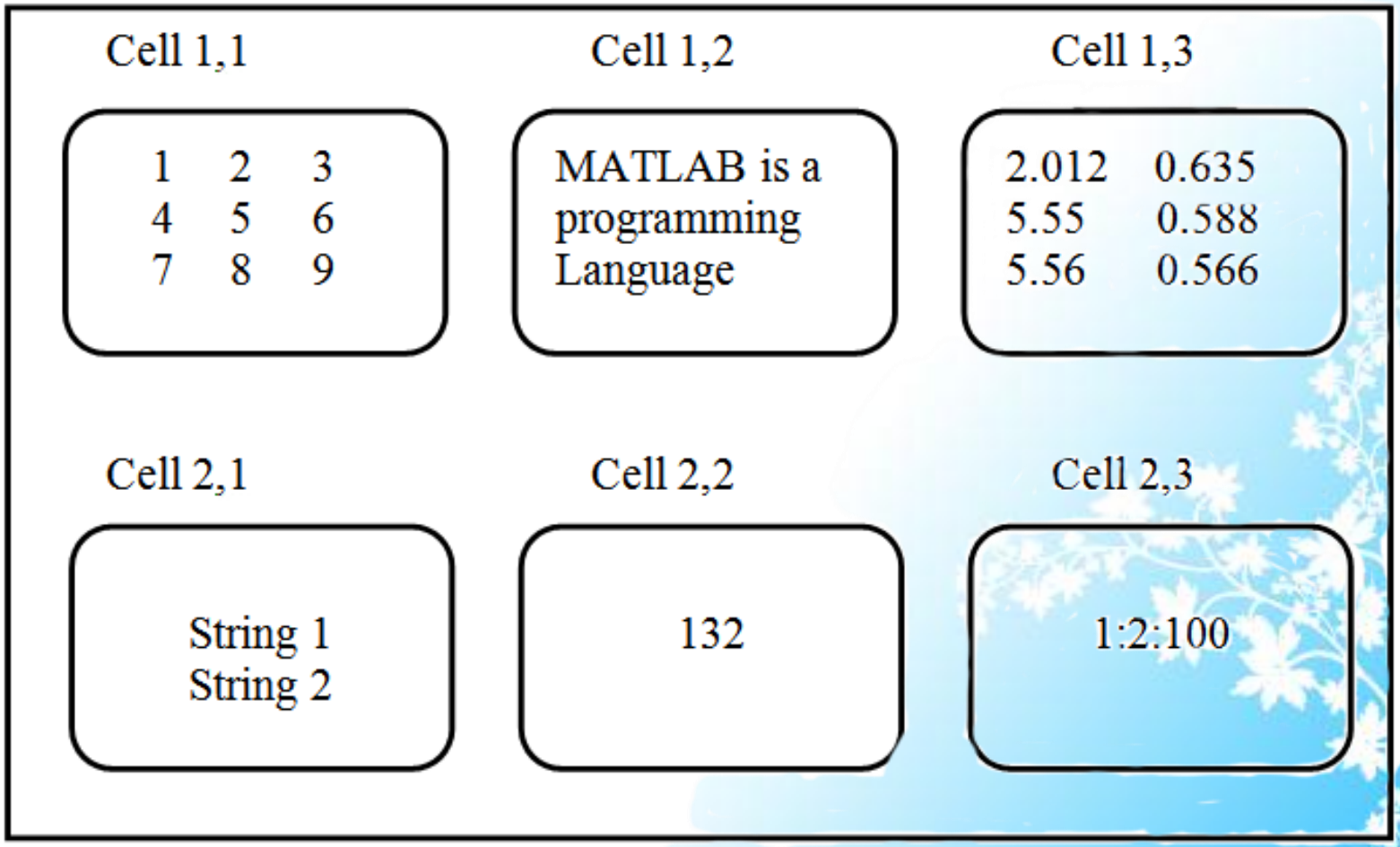
Cell array is the container, which provides a hierarchical storage mechanism for the different types of data. It is MATLAB array and each element of this array is called the *cell*. You can store the data of different types or different dimensions or sizes within a cell of the cell array.

Elements of cell array are enclosed in a pair of curly braces and separated by blank space as shown in the following example:

```
>> B={'MATLAB' 6.5 magic(2)}
B =
 'MATLAB'    [6.5000]    [2x2 double]
```

Cont..

Cell Array:



Cont..

- Creating cell array: you can create or build cell array in two ways:

  1) Using the assignment statement

  2) Pre-allocating the array using cell function

1) Using the assignment statement :

We can create a cell array by using assigning data to individual cells, one cell at a time. Cell index encloses the cell subscripts in parentheses using standard array notation. It also encloses the cell contents on the right side of the assignment statement curly braces "{ }".

Cont..

Example:

```
A(1,1)={[1 2  3; 4 5 6; 7 8 9]};
A(1,2)={'MATLAB is  programming language'};
A(2,1)={132};
A(2,2)={1:2:10};
```

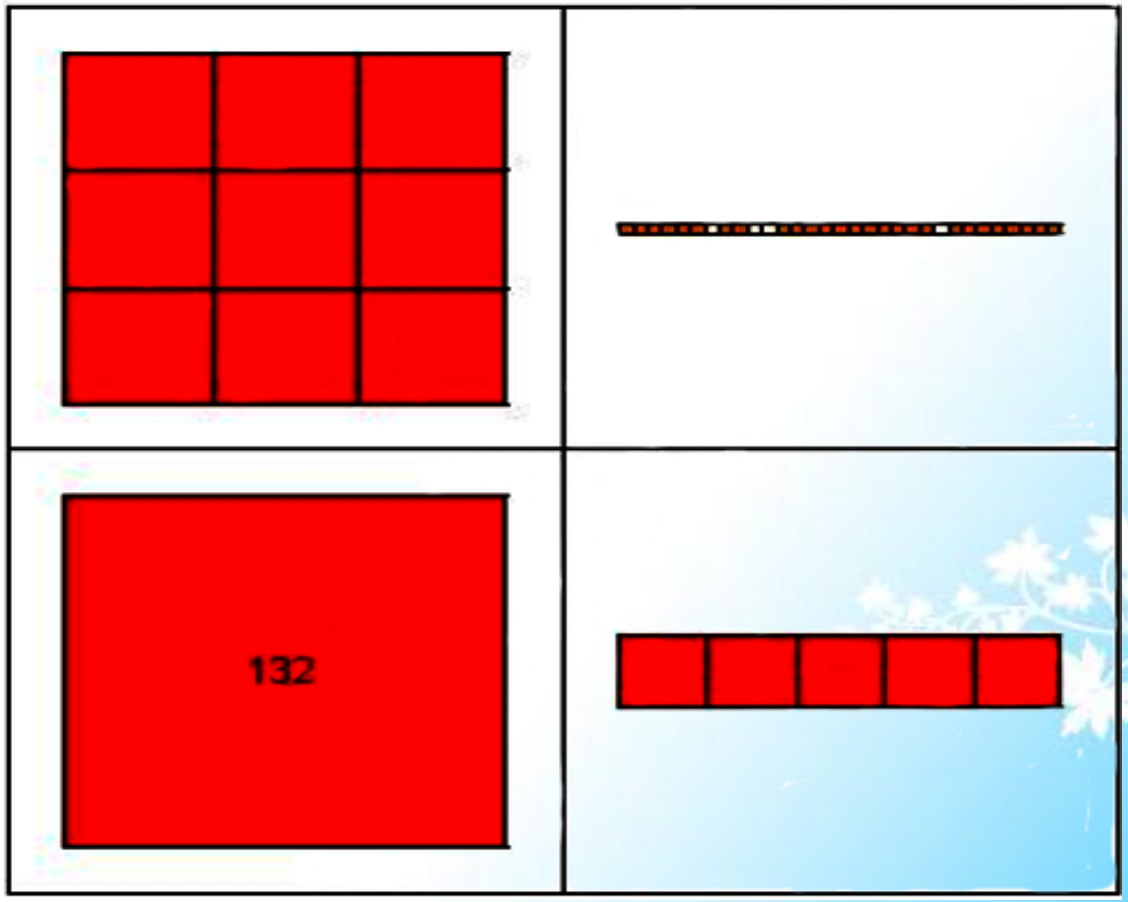
Content indexing encloses the cell subscripts in curly braces by using standard array notation. Specify the cell contents on the right side of the assignment. To display the full content of cell and high-level graphical display of cell architecture use *celldisp* and *cellplot* command respectively. If you assign data to a cell array i.e. outside the dimensions of the current array, MATLAB automatically expands the array to include the sub-scripts you specify.

Cont..

```
>> celldisp(A)

A{1,1} =

     1     2     3
     4     5     6
     7     8     9

A{2,1} =

   132

A{1,2} =

MATLAB is a programming language

A{2,2} =

     1     3     5     7     9
```

Cont..

>> cellplot(A)



Cont..

2) Pre-allocating cell array with cell function:

We can build or create a cell array using *cell* function or command. A cell function allows you to pre-allocate an empty cell array of the specified size.

Example:

B=cell(2,3); % it creates an empty cell array of size 2-by-3.

You access data in a cell array using the same matrix indexing used on other MATLAB matrices and arrays. Based on the obtained or accessed data, we can store these results as either a standard array or new cell array.

Cont..

Accessing cell contents using constants indexing:

Example:

```
>> B(1,1)={'KVK'};
>> B(1,2)={[1 2 3 4 5]};
>> B(2,1)={[2 2 2; 3 3 3;4 4 4]};
>> B(2,2)={0.215}

B =
      'KVK'           [1x5 double]
   [3x3 double]    [    0.2150]
```

Here we can obtain the string by using,

```
>> str=B(1,1)
>> str=B(1,1)
str =
   'KVK'
```

Cont..

Similarly,

```
>> C=B(1,2);
>> D=B{1,2};
>> whos
```

Name	Size	Bytes	Class
B	2x2	366	cell array
C	1x1	100	cell array
D	1x5	40	double array

When we obtain data using the parenthesis MATLAB creates another separate cell array but when we access data using curly brackets MATLAB creates a double array of length of the data.

Cont..

Accessing sub-sets of cell array using cell indexing:

We can use the cell array indexing to assign any set of cells to another variable. For creating or building a new sub-set of cells the colon (:) operator is used.

Example: A is a cell array and B is a subset of A.

Cell (1,1) **10**	*Cell (1,2)* **9**	*Cell (1,3)* **5**
Cell (2,1) **15**	*Cell(2,2)* **2**	*Cell(2,3)* **6**
Cell (3, 1) **5**	*Cell(3,2)* **3**	*Cell(3,3)* **7**

B=A(2:3,2:3)

Cell (1,1) **2**	*Cell(1,2)* **6**
Cell(2,1) **3**	Cell(2,2) **7**

Cont..

Deleting and resizing the cell array:

Use the vector subscripting when deleting a column or row of cells and assign the empty matrix to the dimension. When deleting a cell do not write curly bracket in the assignment statement at all.

```
>> A

A =
    [3x3 double]    [1x31 char  ]
    [       132]    [1x5  double]

>> A(2,:)=[]
A =

    [3x3 double]    [1x31 char]
```

Cont..

Just like other arrays we can resize the cell array by using the *reshape* function. For example

```
>> A=cell(3,4);
>> size(A)
ans =
        4
>> B=reshape(A,12,1);
>> size(B)
ans =
        1
```

Note: The number of cells must remain the same after reshaping and we cannot use the *reshape* function for adding and removing cells in a cell array. Cell array is better than structures when we need to access multiple fields of data with one statement, subsets with separate variables. For applying functions and operators on cells of cell array use the indexing.

Cont..

Some MATLAB cell array functions:

Function Name	Description
cell	Pre-defined cell array structure
Celldisp	Display contents of a cell array
cellplot	Plot structure of a cell array
cellstr	Convert a 2-D character array to a cell array of Strings
char	Convert a cell array of strings to 2-D character array

Structure

Structures are the same as cell arrays; they also store different types of data and organize by using fields. Structures provide a hierarchical storage mechanism for dissimilar type of data. We can access the structure data using the named fields. It is also a MATLAB array with named "Data Container" called fields. The field of structure can contain any kind of data. The MATLAB array and cell array store the data using indexing but the structures store the data using the fields. A structure is an array of structures. Each structure in the array will have identical fields, but the data stored in each field can differ. A single structure is a 1-by-1, structure array. You can build a structure array with any valid size or shape.

Cont..

Example: This example is the structure of employee data:

First field contains the employee no., second field contains the name of the employee, third field contains the basic salary of the employee and forth field contains the official reports.

Creating structures:

Structure array can be created in two ways:

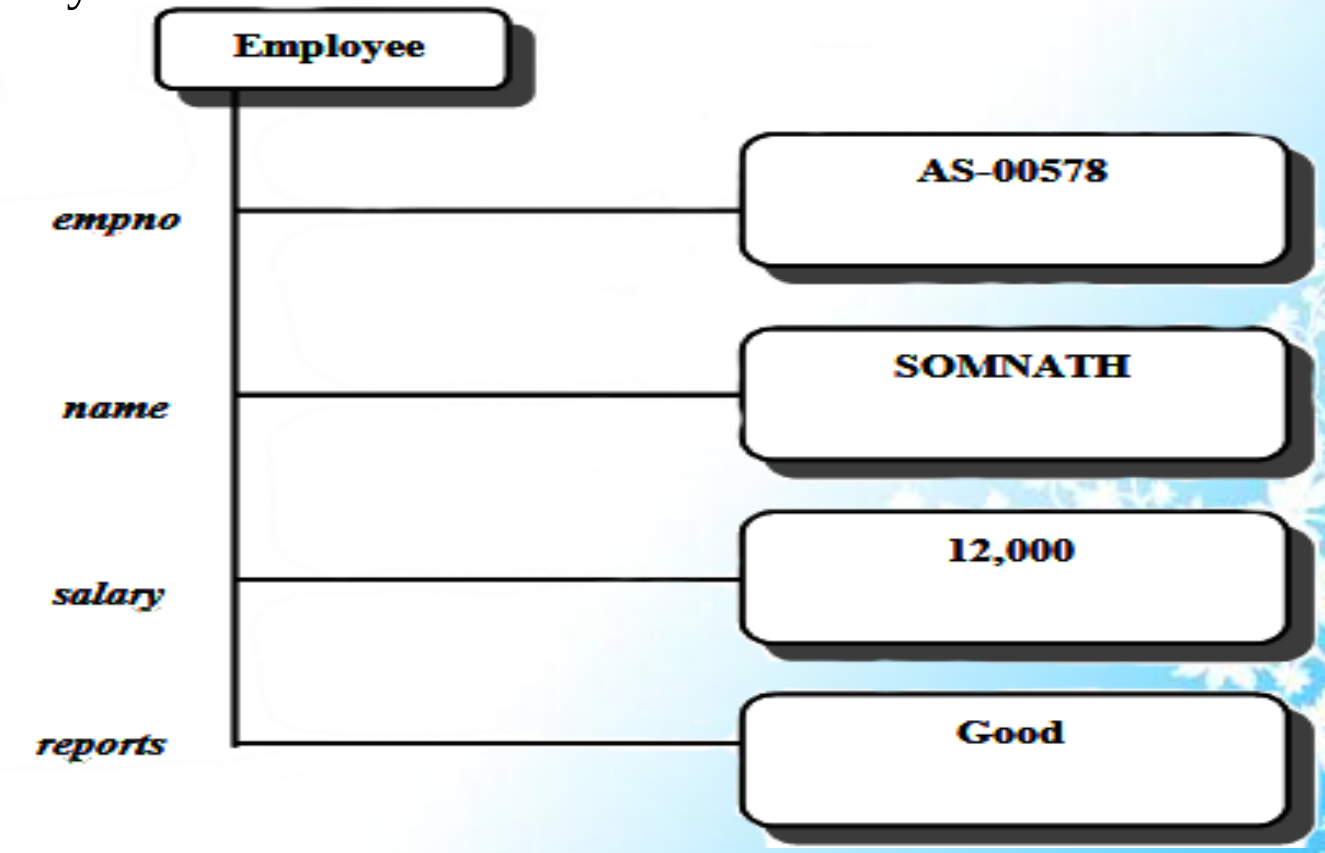
1) Using assignment statement

2) Using the *struct* function

Cont..

Building structure with assignment statement:

You can build a structure simply one field at a time with assignment statements. Each time that data is assigned to a field that field is automatically created.



Cont..

You can build a simple 1-by-1, structure array by assigning the data to individual fields.

Example:

```
>>Employee.empno= 'AS-00578';
>>Employee.name='SOMNATH';
>>Employee.salary=12000;
>>Employee.reports='Good';
```

To expand the structure array add subscripts after the structure name like,

```
>>Employee(2).empno= 'AS-00345';
>>Employee(2).name='RAVI';
>>Employee(2).salary=14500;
>>Employee(2).reports='Excellent';
```

```
>> Employee

Employee =

1x2 struct array with fields:
    empno
    name
    salary
    reports
```

Cont..

```
>> Employee(1)
ans =
        empno: 'AS-00578'
        name: 'SOMNATH'
        salary: 12000
        reports: 'Good'

>> Employee(2)
 ans =
        empno: 'AS-00345'
        name: 'RAVI'
        salary: 14500
        reports: 'Excellent'
```

Note: Employee is a 1-by-2 array, when structure array is more than one element, only the field names are listed, not their contents. The contents of each element can be listed by using element separately in the command window.

Cont..

Building Structures with *struct* function:

The *struct* function allows us to pre-allocate a structure or an array of structures. The syntax of this function is:

str_array=struct('field1',value1, 'field2',value2…)

Where, the arguments are field names and their corresponding values. With this syntax we can initialize every field to the specified value of any MATLAB data construct.

Example:

Employee(3)=struct('empno','AS00848','name','RAMA','salary',12500,'report','Good');

Cont..

Adding fields to Structures:

If a new field name is defined for any element in a structure array the field is automatically added to all of the elements in the array. For example here we add the joining date (joid) field of the Employee structure

```
>>Employee(1).joid=[12 01 1998]
Employee =
1x3 struct array with fields:
    empno
    name
    salary
    report
    joid
```

Cont..

Here, Employee(1).joid has the assigned value. Every other structure in the array also has the joid field, but these fields contain the empty matrix until you explicitly assign a value to them as shown below the Employee(2) and Employee(3) does not have the joid values:

```
>> Employee(1)
ans =
empno: 'AS-00578‘
name: 'SOMNATH‘
salary: 12500
report: 'Good'
joid: [12 1 1998]
```

```
>> Employee(2)
ans =
empno: 'AS-00345'
name: 'RAVI'
salary: 14500
report: 'Excellent'
joid: [ ]
```

```
>> Employee(3)
ans =
empno: 'AS-00848'
name: 'RAMA'
salary: 12500
report: 'Good‘
joid: []
```

Cont..

Accessing Data from Structures:

We can access the value of any field of the structure using the structure array indexing. Similarly like cell array we can access sub-arrays by appending standard subscripts to a structure array name. For example

```
>>myemp=Employee(2:3)
myemp =
1x2 struct array with fields:
    empno
    name
    salary
    report
    joid
```

Cont..

Also we can extract field values for multiple structures at a time, for example

```
>> Names=[Employee.name]
Names =
        SOMNATH   RAVI   RAMA
>> Salary=[Employee.salary]
Salary =
          12000      14500      12500
```

To access a field of a particular structure, include a period (.) after the structure name followed by the field name.

```
>> Name=Employee(2).name
Name =
        RAMA
```

Cont..

To access the elements with field, append the appropriate indexing mechanism to the field name. That is if the field contains an array, use array subscripting,

```
>> year=Employee(1).joid(3)
year =
      1998
```

Also we can access the specified field values by using the setfield and getfield functions. But the direct indexing is usually the most efficient way to assign or retrieve field values.

Cont..

Removing field from the structures:

The *rmfield* function is used to remove the field from structures. The syntax of this function is

Struct1=rmfield(str_array, 'field')

Where str_array is the structure array, field is the field which is to remove, and struct1 is the name of the new structure with that field removed. In this example we can remove the field 'report' from structure array Employee with the following command:

```
>> struct1=rmfield(Employee, 'report')
struct1 =
1x3 struct array with fields:
    empno
    name
    salary
    joid
```

Cont..

Nested Structure :

Each field of a structure array can be of any data type, including a cell array or a structure array. For example, the following statements define a new structure array as a field under Employee to carry information about class of the employee (designation and rank of employee)

```
>> Employee(1).class(1).designation='Production Enginear';
>> Employee(1).class(1).rank=3;
>> Employee(1)
ans =
        empno: 'AS-00578'
        name: 'SOMNATH'
        salary: 12000
        report: 'Good'
        joid: [12 1 1998]
        class: [1x1 struct]
>> Employee(1).class
ans =
    designation : 'Production Enginear'
          rank     : 3
```

Sparse Array

This is a special kind of matrix or array which contains only a significant number of zero valued elements means it allows us to store only nonzero elements of the matrix together with their indices i.e. automatically reduce the computational time by elimination operations on zero element.

Generally in MATLAB it requires the same amount of storage for every element for non-zero or zero element. But in a sparse matrix, MATLAB stores only the nonzero elements and their indices. It will help us to reduce the amount of memory required for large matrices with a high percentage of zero-valued elements for data storage.

Cont..

Example:

```
>> A=2*eye(10)

A =

     2     0     0     0     0     0     0     0     0     0
     0     2     0     0     0     0     0     0     0     0
     0     0     2     0     0     0     0     0     0     0
     0     0     0     2     0     0     0     0     0     0
     0     0     0     0     2     0     0     0     0     0
     0     0     0     0     0     2     0     0     0     0
     0     0     0     0     0     0     2     0     0     0
     0     0     0     0     0     0     0     2     0     0
     0     0     0     0     0     0     0     0     2     0
     0     0     0     0     0     0     0     0     0     2
>> sparse_A=sparse(A)

sparse_A =

   (1,1)        2
   (2,2)        2
   (3,3)        2
   (4,4)        2
   (5,5)        2
   (6,6)        2
   (7,7)        2
   (8,8)        2
   (9,9)        2
   (10,10)      2

>> whos
  Name        Size              Bytes  Class
  A           10x10               800  double array
  sparse_A    10x10               164  double array(sparse)

Grand total is 110 elements using 964 bytes
```

Cont..

Generating Sparse Matrices:

MATLAB never creates sparse matrices automatically until we determine them. MATLAB can create or convert a full matrix into sparse matrices by using *sparse* function or directly using the sparse matrices with the functions *speye, sprand, sprandn* etc.

```
>> S=speye(6)
 S =
   (1,1)        1
   (2,2)        1
   (3,3)        1
   (4,4)        1
   (5,5)        1
   (6,6)        1
>> S1=full(S)
 S1 =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     1
```

We can assign and delete the values of sparse matrices by using the assignment statement like the ordinary matrices manipulation. For example

```
>> S(5,3)=13

S =

   (1,1)        1
   (2,2)        1
   (3,3)        1
   (5,3)       13
   (4,4)        1
   (5,5)        1
   (6,6)        1
```

Cont..

Example 1:

```
% Write a program to create a structure called showroom consisting of different
% vehicle names and their appropriate prices
showroom(1).vehicle='Maruti';
showroom(1).price= '5 lacks';
showroom(2).vehicle='Zen';
showroom(2).price='6 lacks';
showroom(3).vehicle='Passian +';
showroom(3).price='46 k';
```

Output:

```
>> showroom
showroom =
1x3 struct array with fields:
                                vehicle
                                 price
```

```
>> showroom.price
    ans =
            5 lacks
    ans =
            6 lacks
    ans =
            46 k
    >> showroom.vehicle
    ans =
            Maruti
    ans =
            Zen
    ans =
            Passian +
    >>
    showroom(1).vehicle,showroom(1).pric
    e
    ans =
            Maruti
    ans =
            5 lacks
```

Summary

Cell Array is the container, which provides a hierarchical storage mechanism for different types of data. You can store the data of different types or different dimensions or size within a cell of the cell array. Cell array can create or build in two different ways: like: Using the assignment statement, Pre-allocating the array using cell function. Structures provide a hierarchical storage mechanism for dissimilar types of data. Structure array can be created in two ways: Using assignment statement and using the *struct* function. . Sparse Array is a special kind of matrix or array which contains only a significant number of zero valued elements means it allows us to store only the nonzero elements of the matrix together with their indices i.e. automatically reduce the computational time by elimination of operations on zero element. MATLAB can create or convert a full matrix into sparse matrices by using *sparse* function or directly using the sparse matrices with the functions *speye*, *sprand*, *sprandn* etc.

Thank You

Exercises

1. Write a program to create a cell array named, as result consists of role no. of 10 students in first cell and name of these in second cell. Extend code of this program to add third cell in this cell array to feed the result of the student. Further write a code to access subset (includes only role no. and result) of this cell array using cell indexing.
2. Write a program to create a cell array that consists of 3 x 4 and 4 x 5 two 2-D matrices and 'MATLAB also support for object orientated' as third element. Display contents of this cell array using assignment statement.
3.