Unit 6:
Data Structures (II)

Matrices

- A two dimmension matrix can be seen as a group of vectors with the **same lengths** and the **same type of elements**.

```
A(1,:) = [1 2 3; 4 5 6; 7 8 9];
```

or

```
A(1,:) = [1 2 3];
A(2,:) = [4 5 6];
A(3,:) = [7 8 9];
```

A

1	2	3
4	5	6
7	8	9

Matrices

- These are not valid matrices, as they have rows with different lengths or different types of elements

A

1	4	
2	5	8
3		

B

1	4	7
2	c	8
A	6	9.2

Two dimension matrices

1. Creation

   varMatrix = []; varMatrix = 1;

   varMatrix = [1 2; 3 4]; varMatrix = zeros (2,5);

   varMatrix = [varVector1; varVector2; varVector3];

   …

2. Use

   - Read an element: varMatrix(row, column);
   - Read a portion: varMatrix(row_ini:row:end, col_ini:col_end);
   - Save/modify an element: varMatrix(row, column) = value;
     varMatrix(position) = value;
   - Delete a column: varMatrix(:,column) = [];
   - Delete a row: varMatrix(row,:) = [];

Two dimension matrices

- *Useful functions:*
  - *size* - Returns the dimensions of a matrix

```
A = [1 2 3; 4 5 6];
size(A)
>> 2   3
```

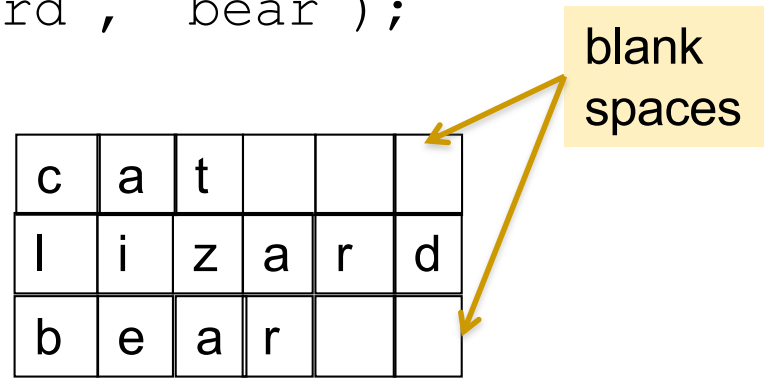
  - *length* - Length of vector or largest matrix dimension
  - *numel* - Number of elements in the matrix
  - *isempty* - Determine whether the matrix is empty

Two dimension matrices

- Useful functions:
  - ***sarr = char(s1, s2, s3,..)***

forms a matrix of characters containing the string s1, s2, s3… as rows. Automatically pads each string with blanks in order to form a valid matrix.

```
sarr = char('cat', 'lizard', 'bear');
```



Cells Arrays

- A cell array is a vector or matrix of ***cells*** , where each cell can contain **data which belongs to any valid MATLAB type**, and can have **any valid array dimensions**.

- To create a cell array use the symbols *{ }*

```
varCell = { 'a'  3 'chain' 0.567 [1 2]}
```

varCell

{ 'a' }	{ 3 }	{ 'chain' }	{ 0.567 }	{ [1 2] }

Cells Arrays

1. Creation

Before creation you must clear the variable using `clear varCell`

varCell = {A B C}; where A, B, C are variables of any type

varCell = { 1 'a' 'chain' '0.4' [1 2 3]};

varCell = { 'Hello' 'Goodbye' 'See you later' };

…

Cells Arrays

varCell

{'a'}	{3}	{'chain'}	{0.567}	{[1 2]}

2. Use
   - Read **the content of a cell**: varCell**{**position**}**;

     ```
     newVar = varCell{1}
     ```
     -> newVar is now 'a'

   - Save/modify an element: varCell**{**position**}** = value

     ```
     varCell {1} = 'See you later';
     ```

   - Delete a cell: varCell**(**position**)** = [ ];

   - Cell Arrays of more than one dimension:

     varCell{row, column} = value;

Be careful!!!

Exercise

- Write a program that asks the user to introduce words until he/she introduces the word *end*. Then the program prints all the words introduced

Example:

Introduce a word: cat
Introduce a word: mouse
Introduce a word: lizard
Introduce a word: end
You have introduced the following words:
cat
mouse
lizard

Exercise

words

{'cat'}	{'mouse'}	{'lizard'}

```
clear words;
cont = 0;
sword = input('Introduce a word: ', 's');
while (strcmp(vword, 'end') ~=1)
    cont = cont +1;
    words{cont} = sword;
    sword = input('Introduce a word: ', 's');
end;
disp('You have introduced the following words: ');
for i=1:cont
    fprintf('\n %s',words{i});
end;
```

Exercise

- Write a program that asks the user to introduce names and ages until he or she introduces an empty name. Next it prints them all following the order of introduction. Use a cell array to store the information

Example:

Introduce a name: Pedro
Introduce an age: 20
Introduce a name: Ana
Introduce an age: 18
Introduce a name: Elena
Introduce a name: 19
Introduce a name:
You've finished introducing names.
Names introduced:
Pedro: 20 years old
Ana: 18 years old
Elena: 19 years old

cellstudents

{'Pedro'}
{20}
{'Ana'}
{18}
{'Elena'}
{19}

Exercise

```
clear cellstudents;
cont = 0;
varName = input('Introduce a name: ', 's');
while (isempty(varName) == 0)
    varAge = input('Introduce an age: ');
    cont = cont + 1;
    cellstudents{cont} = varName;
    cont = cont + 1;
    cellstudents{cont} = varAge;
    varName = input('Introduce a name: ', 's');
end;
disp('You finished introducing names');
disp('Names introduced');
for i=1:2:cont
    j = i +1;
    fprintf('\n %s: %d years old ', cellstudents{i}, cellstudents{j});
end;
```

cellstudents

{'Pedro'}
{20}
{'Ana'}
{18}
{'Elena'}
{19}

Exercise

- Write a program which asks the user to introduce names and ages until he or she introduces an empty name. Next it prints them all following the order of introduction. Use a cell array of two dimensions (a matrix) to store the information

Example:

Introduce a name: Pedro
Introduce an age: 20
Introduce a name: Ana
Introduce an age: 18
Introduce a name: Elena
Introduce a name: 19
Introduce a name:
You've finished introducing names.
Names introduced:
Pedro: 20 years old
Ana: 18 years old
Elena: 19 years old

cellstudents

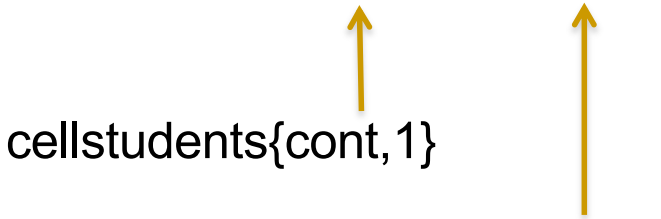
{'Pedro'}	{20}
{'Ana'}	{18}
{'Elena'}	{19}

Exercise

```
clear cellstudents;
cont = 0;
varName = input('Introduce a name: ', 's');
while (isempty(varName) ==0)
    varAge = input('Introduce an age: ');
    cont = cont + 1;
    cellstudents{cont,1} = varName;
    cellstudents{cont,2} = varAge;
    varName = input('Introduce a name: ', 's');
end;
disp('You finished introducing names');
disp('Names introduced');
for i=1:cont
    fprintf('\n %s: %d years old ', cellstudents{i,1}, cellstudents{i,2});
end;
```

cellstudents

{'Pedro'}	{20}
{'Ana'}	{18}
{'Elena'}	{19}



Structures

- Sometimes we need to handle many different types of information together:
  - Example: the information about a student might include his or her name, surename, NIA, age, address..

- So far we have seen two ways of working around these situations:
  1. Using different variables, vectors or matrices for each type of data

```
varStName     = 'Ana'
varStNIA      = 10003456
varStCourse   = 'Programming'
```

  2. Using cellArrays

```
cellStudent = {'Ana', 10003456, 'Programming'}
```

Structures

- Structures give us a way to "combine" multiple types of information under a single variable.
  - We could do this using cells, or variables, vectors or matrices for each type of data…
  - … but structures allow us to use **Human Readable** descriptions for our data

- Structures are multidimensional MATLAB arrays with elements accessed by **textual *field designators***.
  - You can think about them as variables that are composed of other variables. We call those subvariables “fields”:

```
variableName.fieldName
```

Structures

- Example: Create a structure to store information about a student

```
student.name   = 'Pedro';
student.nia    = 10004567;
student.course = 'Programming';
>> student
            name: 'Pedro'
            nia:  10004567
            course: 'Programming'
```

"student" is a variable of type structure that is composed of three fields: "name", "nia" and "course"

Structures

2. Creation

- Field by Field

```
variableName.fieldName = value;
```

Example:

```
student.name   = 'Pedro';
student.nia    = 10004567;
student.course = 'Programming';
```

- Using the command *struct*

```
variableName = struct(
                    fieldName1,fieldvalue1,
                    fieldName2, fieldvalue2,…);
```

Example:

```
student = struct ('name', 'Pedro', 'nia', 10004567,
'grade','biomedical');
```

Structures

2. Use
   - Access an element: *varName.field_name;*
   - Display all the content: *varName* *(only use this one in the command line not in a program)*
   - Save/modify an element: *varName.field_name = value;*
   - Delete a field: *varName = rmfield(varName,'field_name');*

Exercise

- Write a program which asks the user to introduce the name, age, and the job of an employee, and then prints all the information. Use a structure to store the information

Example:

Introduce the name: Pedro
Introduce the age : 24
Introduce the job: Engineer
Worker's information: Pedro, 24 years old, Engineer

Exercise

```
worker.name = input (’Introduce the name: ','s');
worker.age = input (‘Introduce the age: ‘);
worker.job = input (‘Introduce the job);
fprintf(’\n Worker information: %s, %d years old,%s.', worker.name, worker.age,
worker.nickname);
```

Vectors/Matrices of Structures

- You can create **vectors or matrices of structures** in the same way as you create vectors or matrices of numbers, characters or any other type of data.

You can image the information of a vector of a vector of structure as organize like this in memory

vstudents

name = ‘Pedro’ nia = 10004567 course = ‘Biomedical’
name = ‘Alberto’ nia = 10004666 course = ‘Computer Science’
name = ‘Juan’ nia = 10004688 course = ‘Maths’

Vectors/Matrices of Structures

- To create the vector simply introduce the values of each field of the structure stored in each position of the vector.

```
vectorname(position).fieldname = value;
```

  - Example:

```
student(1).name   = 'Pedro';
student(1).nia    = 10004567;
student(1).course = 'Biomedical';
student(2).name   = 'Alberto';
student(2).nia    = 10004666;
student(2).course = 'Computer Science';
```

vstudents

name = 'Pedro' nia = 10004567 course = 'Biomedical'
name = 'Alberto' nia = 10004666 course = 'Computer'

Vectors/Matrices of Structures

- To access the structure stored in a position of a vector

```
vectorname(position)
```

- To access a field of a structure stored in a position of a vector

```
vectorname(position).fieldname
```

  - Example:

```
>> student(1)
name: 'Pedro'
nia:  10004567
course: 'Biomedical'
>> student(1).course
        Biomedical
```

Exercise

- Write a program which asks the user to introduce the names, ages and jobs of different employee and then prints all the information.

Example:

Introduce the name: Pedro
Introduce the age : 24
Introduce the job: Engineer
Do you want to introduce information about another emploee (Y/N)? Y
Introduce the name: Ana
Introduce the age : 27
Introduce the job: Doctor
Do you want to introduce information about another emploee (Y/N)? Y
Introduce the name: Juan
Introduce the age : 32
Introduce the job: Programmer
Do you want to introduce information about another emploee (Y/N)? N
Worker 1: Pedro, 24 years old, Engineer
Worker 2: Ana, 27 years old, Doctor
Worker 3: Juan, 32 years old, Programmer

Exercise

```
vworkers   = [ ];
cont       = 0;
cContinue  = 'Y';
while (cContinue == 'Y')
    cont = cont + 1;
    vworkers(cont).name = input ('Introduce the name: ','s');
    vworkers(cont).age  = input ('Introduce the age: ');
    vworkers(cont).job  = input ('Introduce the job: ','s');
    cContinue = input('Do you want to introduce information about another emploee (Y/N)?','s');
end;
for i=1:cont
   fprintf('\n Worker %d: %s, %d years old, %s.', i, vworkers(i).name, vworkers(i).age, vworkers(i).job);
end;
```

Exercise

- Modify the previous program so at the end it asks the user to introduce a number at prints the information of the corresponding worker following the order of introduction.

Example (continuation):

. . .

Introduce the name: Juan
Introduce the age : 32
Introduce the job: Programmer
Do you want to introduce information about another emploee (Y/N)? N
Introduce a number: 2
Worker 2: Ana, 27 years old, Doctor

Exercise

```
vworkers  = [ ];
cont      = 0;
cContinue = 'Y';
while (cContinue == 'Y')
    cont = cont + 1;
    vworkers(cont).name = input ('Introduce the name: ','s');
    vworkers(cont).age  = input ('Introduce the age: ');
    vworkers(cont).job  = input ('Introduce the job: ','s');
    cContinue = input('Do you want to introduce information about another emploee (Y/N)?','s');
end;
number = input('Introduce a number: ');
fprintf('\n Worker %d: %s, %d years old, %s.', number, vworkers(i).name, vworkers(i).age,
vworkers(i).job);
```

Exercise

- Write a program which asks the user to introduce coordinates *x* and *y* of points. Next the program prints the information of all the points introduced one after the other.

Exercise

Example of execution:

Introduce the x coordinates; 1.5
Introduce the y coordinates; 2.5
Do you want to introduce more points (Y/N)? Y
Introduce the x coordinates; 3
Introduce the y coordinates; 5.12
Do you want to introduce more points (Y/N)? Y
Introduce the x coordinates; 2
Introduce the y coordinates; 2
Do you want to introduce more points (Y/N)? N
The coordenates of the points are:
Point 1: (1.50, 2.50)
Point 1: (3.00, 5.12)
Point 3: (2.00, 2.00)

Exercise

```
% Introduction of the points

count = 0;
more  = 'Y';
while more == 'Y'
    count = count + 1;
    point(count).x = input('Introduce the x coordinates: ');
    point(count).y = input('Introduce the y coordinates: ');
    more = input('Do you want to introduce more points (Y/N)? ','s');
end;
disp ('The coordinates of the points are: ');
for i = 1:count
            fprintf('\n Point %d: (%.2f, %.2f)', i, point(i).x, point(i).y);
end;
```

Exercise

- Modify the previous program so that after introducing the points the user is asked to introduce a number. The program will print on screen the distances from the point selected by the user (according to their order of introduction) to the rest of the points. Finally, it displays which is the furthest point to the one selected by the user.

Exercise

Example of execution:

Introduce the x coordinates; 1.5
Introduce the y coordinates; 2.5
Do you want to introduce more points (Y/N)? Y
Introduce the x coordinates; 3
Introduce the y coordinates; 5.12
Do you want to introduce more points (Y/N)? Y
Introduce the x coordinates; 2
Introduce the y coordinates; 2
Do you want to introduce more points (Y/N)? N
Introduce the number of one of the points: 2
The distances to the point (3.00,5.12) are:
Point 1: distance 2.12
Point 3: distance 1.41
The closest point is the point (2.00,2.00)