

# Matlab Data Structures

Dr. Antonio A. Trani  
Professor  
Dept. of Civil and Environmental Engineering

# Why Learn Data Structures?

- Engineers need to manipulate large amounts of data
- Data sometimes comes in a variety of formats
- Data is both numeric and character or “string” data
- Matlab has two important structures that you should be familiar with:
  - Struct arrays
  - Cell arrays

# Recall: Reading Data Files Using the Textscan Command

- Using the **Textscan** Command
- Here is a sample script to read a text file containing data on bridges of the world

```

fid = fopen('bridges_of_the_world_short')
readHeader = textscan(fid, '%s', 4, 'delimiter', '|');
readData = textscan(fid, '%s %s %f %f');
fclose(fid);

```

# Data File (bridges\_of\_the\_world)

Name	Country	Completed	Length (m)
Mackinac	United-States	1957	8038
Xiasha	China	1991	8230
Virginia-Dare-Memorial	United-States	2002	8369
General-Rafael-Urdaneta	Venezuela	1962	8678
Sunshine-Skyway	United-States	1987	8851
Twin-Span	United-States	1960	8851
Wuhu-Yangtze-River	China	2000	10020
Third-Mainland	Nigeria	1991	10500
Seven-Mile	United-States	1982	10887
San-Mateo-Hayward	United-States	1967	11265
Leziria-Bridge	Portugal	2007	11670
Confederation	Canada	1997	12900
Rio-Niterol	Brazil	1974	13290
Kam-Sheung	Hong Kong	2003	13400
Penang	Malaysia	1985	13500
Vasco-da-Gama	Portugal	1998	17185
Bonnet-Carre-Spillway	United-States	1960	17702
Chesapeake-Bay-Bridge-Tunnel	United-States	1964	24140
Tianjin-Binhai	China	2003	25800
Atchafalaya-Swamp-Freeway	United-States	1973	29290
Donghai	China	2005	32500
Manchac-Swamp	United-States	1970	36710
Lake-Pontchartrain-Causeway	United-States	1956	38422

Header

Data

# Explanations of the Matlab Script

```
fid = fopen('bridges_of_the_world_short')
```

- **fid** - file ID assigned by Matlab
- **fopen** - “opens” (or reads) the text file called ‘bridges\_of\_the\_world’

```
readHeader = textscan(fid, '%s', 4, 'delimiter', '|');
```

- variable **readHeader** will store the contents of the first row in the file (‘bridges\_of\_the\_world’)
- **textscan** reads the first row of the file using ‘%s’,4 (four string variables) with ‘delimiter’ = ‘|’

Name	Country	Completed	Length (m)
Mackinac	United-States	1957	8038
Xiasha	China	1991	8230
Virginia-Dare-Memorial	United-States	2002	8369

# Explanations of the Matlab Script

```
readData = textscan(fid, '%s %s %f %f');
```

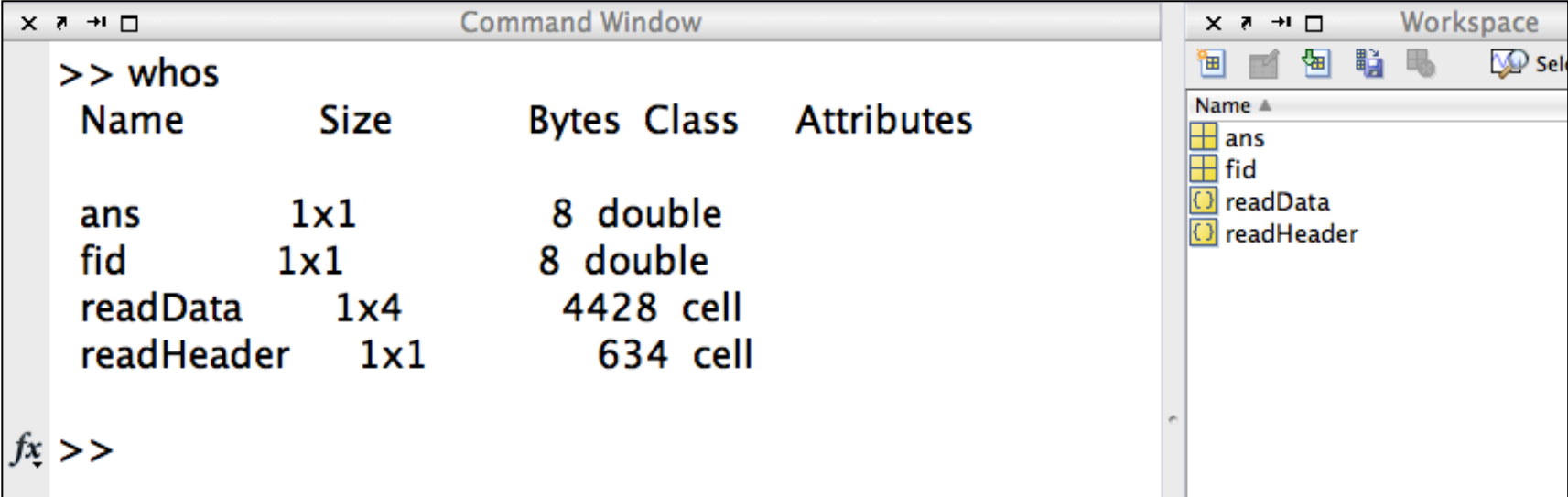
- variable readData will store the contents of the information starting in the second row (until the end) in the file ('bridges\_of\_the\_world')
- textscan reads the row data using '%s %s' two string variables and two '%f %f' numerical variables (f stands for floating point)

```
fclose(fid);
```

- fclose(fid) closes the file (fid) opened at the beginning of the script

Name	Country	Completed	Length (m)
Mackinac	United-States	1957	8038
Xiasha	China	1991	8230
Virginia-Dare-Memorial	United-States	2002	8369

# What is Produced by the Matlab Script?



The screenshot shows the MATLAB Command Window and Workspace. The Command Window displays the output of the 'whos' command, which lists the variables in the workspace. The Workspace window shows the same variables: 'ans', 'fid', 'readData', and 'readHeader'.

```
>> whos
Name      Size      Bytes Class      Attributes

ans       1x1        8 double
fid       1x1        8 double
readData  1x4       4428 cell
readHeader 1x1       634 cell

fx >>
```

Name	Size	Bytes	Class	Attributes
ans	1x1	8	double	
fid	1x1	8	double	
readData	1x4	4428	cell	
readHeader	1x1	634	cell	

- Four variables (2 are temporary - ans and fid)
- Two variables with the information in the file (*readHeader* and *readData*)
- Both variables are **cell arrays**

# What is a Cell Array?

- A special structure in Matlab to store dissimilar data types (i.e., strings and numeric data)

```
>> readData
```

```
readData = {14x1 cell} {14x1 cell} [13x1 double] [13x1 double]
```

Bridge Name

Country

Year  
Completed

Length (m)



# Addressing the Contents of a Cell Array

- Cell arrays are referenced using curly brackets (first) then using standard brackets - to address individual elements of the cell array
- `readData{1}` references the first column of the array

```
>> readData{1}
ans =
'Mackinac'
'Xiasha'
'Virginia-Dare-Memorial'
'General-Rafael-Urdaneta'
'Sunshine-Skyway'
'Twin-Span'
'Wuhu-Yangtze-River'
'Third-Mainland'
'Seven-Mile'
'San-Mateo-Hayward'
'Leziria-Bridge'
'Confederation'
'Rio-Niterol'
'Kam-Sheung'
```



# Addressing the Contents of a Cell Array

- Cell arrays are referenced using curly brackets (first) then using standard brackets - to address individual elements of the cell array
- `readData{1}(3,1)` references the third row element of the cell array

```
Command Window
>> readData{1}(3,1)
ans =
    'Virginia-Dare-Memorial'
fx >>
```

```
>> readData{1}
ans =
    'Mackinac'
    'Xiasha'
    'Virginia-Dare-Memorial'
    'General-Rafael-Urdaneta'
    'Sunshine-Skyway'
    'Twin-Span'
    'Wuhu-Yangtze-River'
    'Third-Mainland'
    'Seven-Mile'
    'San-Mateo-Hayward'
    'Leziria-Bridge'
    'Confederation'
    'Rio-Niterol'
    'Kam-Sheung'
```

# Addressing the Contents of a Cell Array

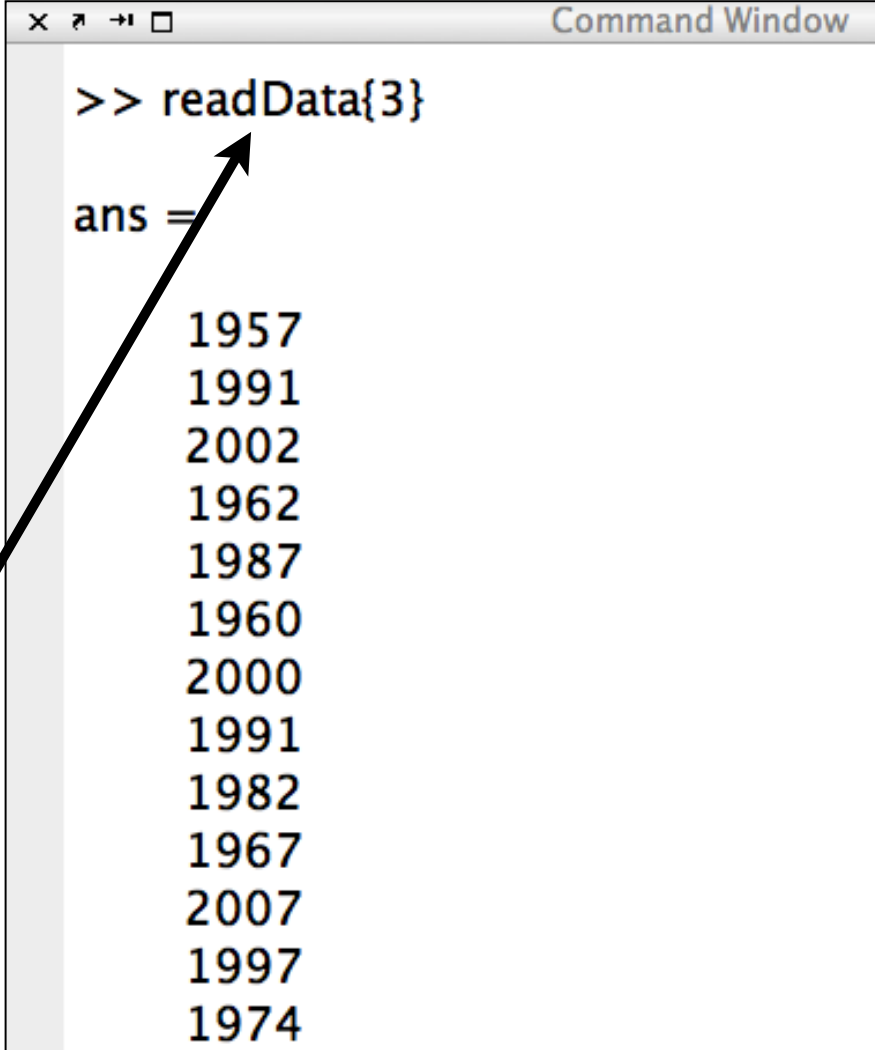
- Cell arrays are referenced using curly brackets (first) then using standard brackets - to address individual elements of the cell array
- `readData{1}(3:5,1)` references the third, fourth and fifth row elements of the cell array

```
>> readData{1}(3:5,1)
ans =
'Virginia-Dare-Memorial'
'General-Rafael-Urdaneta'
'Sunshine-Skyway'
fx >>
```

```
>> readData{1}
ans =
'Mackinac'
'Xiasha'
'Virginia-Dare-Memorial'
'General-Rafael-Urdaneta'
'Sunshine-Skyway'
'Twin-Span'
'Wuhu-Yangtze-River'
'Third-Mainland'
'Seven-Mile'
'San-Mateo-Hayward'
'Leziria-Bridge'
'Confederation'
'Rio-Niterol'
'Kam-Sheung'
```

# Addressing the Contents of a Cell Array

- Cell arrays are referenced using curly brackets (first) then using standard brackets - to address individual elements of the cell array
- **readData{3}** references all the elements of the third column of the cell array

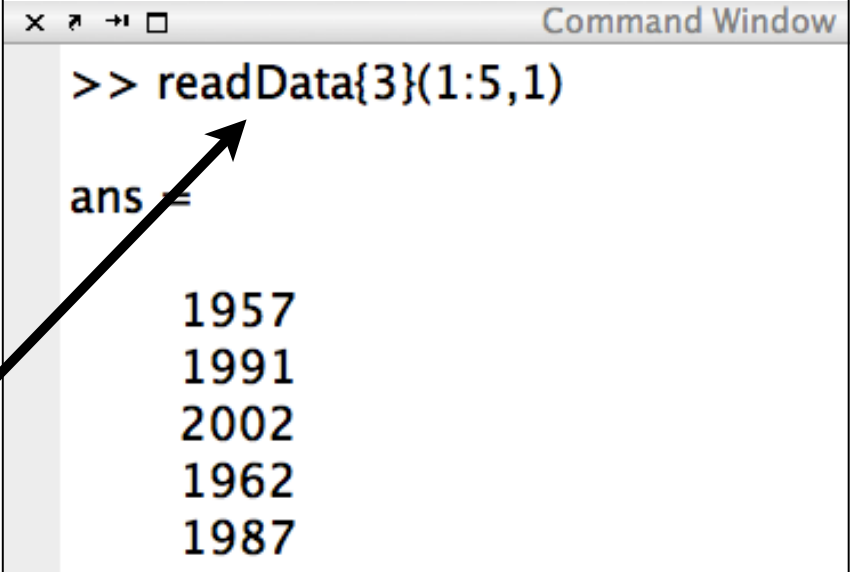


```
Command Window
>> readData{3}
ans =
    1957
    1991
    2002
    1962
    1987
    1960
    2000
    1991
    1982
    1967
    2007
    1997
    1974
```

An arrow points from the **readData{3}** code in the list to the corresponding output in the Command Window.

# Addressing the Contents of a Cell Array

- Cell arrays are referenced using curly brackets (first) then using standard brackets - to address individual elements of the cell array
- **readData{3}(1:5,1)** references the first five row elements of the third column of the cell array



```
Command Window
>> readData{3}(1:5,1)
ans =
    1957
    1991
    2002
    1962
    1987
```

# Manipulating Data inside Cell Arrays

- Now that we have the data try a few things:
  - Question 1: Suppose we want to know how many of the bridges of the World happen to be in the United States
  - Question 2: Suppose that we wanted to know the average bridge length of bridges in China

Question 1: Suppose we want to know how many of the bridges of the World happen to be in the United States

- Use the string comparison function in Matlab

**strcmp**

- Function that compares a string with an array of strings (or cell array) and outputs the position of the array where a match occurs

`strcmp(readData{2}, 'United-States')`

- Here we compare the elements of cell array `readData{2}` with the string “United-States”

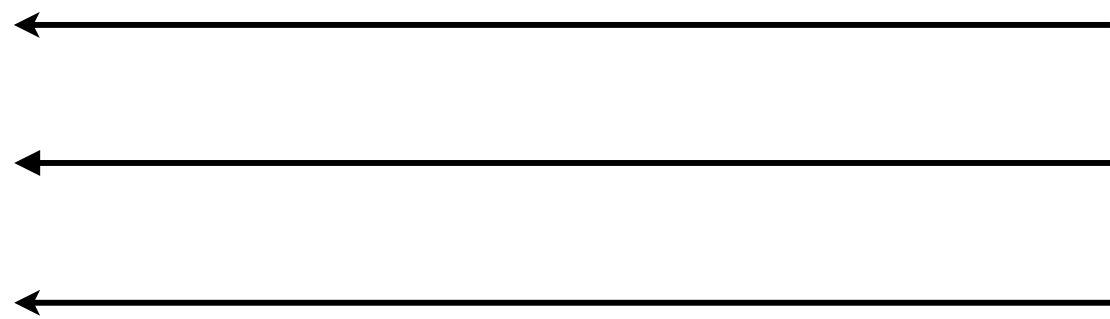
Question 1: Suppose we want to know how many of the bridges of the World happen to be in the United States

```
>> matches = strcmp(readData{2}, 'United-States')
```

Original data  
readData{2}

matches =

- 1
- 0
- 1
- 0
- 1
- 1
- 0
- 0
- 1
- 1



- 'United-States'
- 'China'
- 'United-States'
- 'Venezuela'
- 'United-States'
- 'United-States'
- 'China'
- 'Nigeria'
- 'United-States'
- 'United-States'

Elements of readData{2} that match the word 'United-States' are assigned a one, otherwise a zero



Question 1: Suppose we want to know how many of the bridges of the World happen to be in the United States

```
>> matches = strcmp(readData{2}, 'United-States')
```

matches =

1  
0  
1  
0  
1  
1  
0  
0  
1  
1

Variable that contains indices of array that match

String Comparison Function in Matlab

Array that we want to match

String to Match

Question 1: Suppose we want to know how many of the bridges of the World happen to be in the United States

```
>> sum(matches)

ans =

    11
```

To get the number of bridges we just sum the instances variable 'Matches'

For this example, there are 11 bridges that are listed under United-States

Question 2: Suppose that we wanted to know the average bridge length of bridges in China

- This tells us the number of bridges in China

```
>> chineseBridges = strcmp(readData{2},'China')
```

```
chineseBridges =
```

```
0
1
0
0
0
0
0
1
0
```

Variable “chineseBridges” contains the indices of bridges located in China (partially shown)

Question 2: Suppose that we wanted to know the average bridge length of bridges in China

- Create a new variable “lengthOfbridgesInChina” to extract the lengths of all Chinese Bridges

```
>> lengthOfBridgesInChina=readData{4}(chineseBridges)
```

```
lengthOfBridgesInChina =
```

```
8230
10020
25800
32500
```

Variable “lengthOfbridgesInChina” contains the lengths of the bridges in China found in the list (partially shown)

# Observe What is Going On

- The array (or matrix) “chineseBridges” is an index matrix with zeros or ones
- The array “chineseBridges” acts as a pointer variable for other computations (a variable used to designate the positions of interest of the original array)
- To add the lengths of the bridges in China then just add the lengths of the individual bridges found in previous step

Question 2: Suppose that we wanted to know the average bridge length of bridges in China

- Create a new variable “totalLenghtOfBridgesInChina” to estimate the total length of bridges in China

```
>> totalLenghtOfBridgesInChina=sum(lengthOfBridgesInChina)
```

```
totalLenghtOfBridgesInChina =
```

```
76550
```

A total of 76,550 meters of bridges are found in China  
The average bridge length is then 17,138 meters

## Try the Following in Class

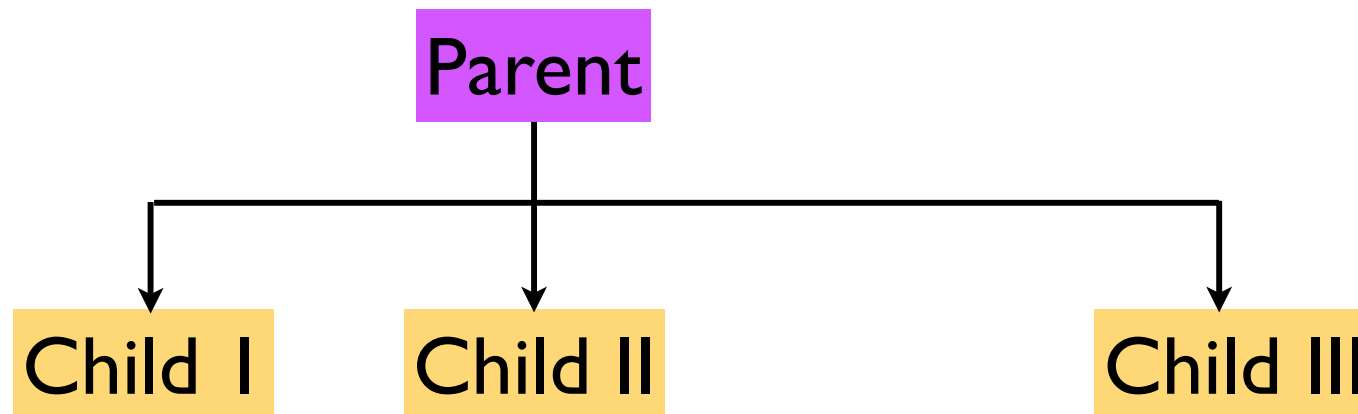
- Find the number of bridges in Portugal
- Find the number of miles of bridges in Portugal
- Find the oldest bridge in Portugal

# Matlab Structured Arrays



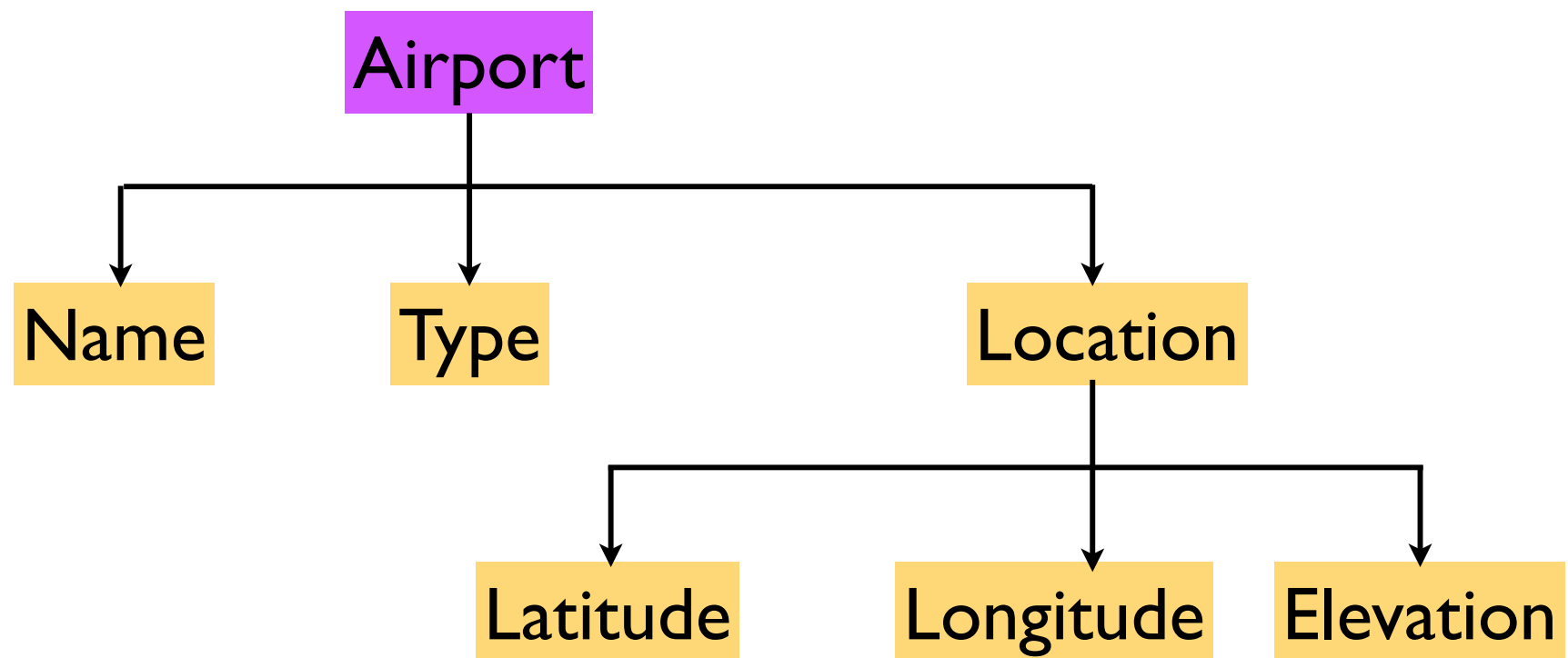
# What is a Structured Array?

- Another Matlab data structure that can store dissimilar information (i.e., strings, numerics)
- A structured file (or struct file) is defined a parent-child relationship
- The parent structure contains high-level information about the data set
- “Children” branches contain detail information related to the parent



# A Sample Structured Array

- An engineer wants to store data about various airports in a Matlab struct file
- The data structure is shown below
- “Airport” has “children” (like name, type) and some children have “grand children” (i.e., location)



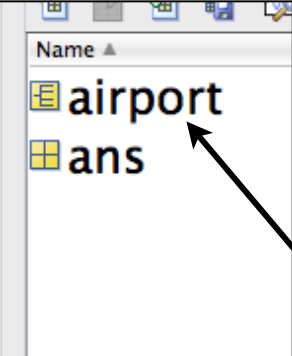
# Sample Struct Array (Airport)

- The struct file “airport” has 3 instances (i.e., records)
- Each instance represents a record of a distinct airport

```
>> airport

airport =

1x3 struct array with fields:
    name
    type
    location
```



The screenshot shows the MATLAB workspace with two variables: 'airport' and 'ans'. An arrow points from the 'airport' variable in the workspace to the text box on the right.

Matlab tells you airport is a struct array

Matlab tells you that there are three children

# Inspecting the Contents of Struct Array Airport

- Instances of the struct array are called directly like any other array in Matlab
- Type “**airport(1)**” at the command line to query the first instance of struct array airport

```
>> airport(1)  
  
ans =  
  
    name: [1x17 char]  
    type: 'Public'  
 location: [1x1 struct]
```

Matlab tells you more  
about the data  
structure  
inside each child  
of airport

# Inspecting the Contents of Struct Array Airport

- Check the contents of `airport(1).name`

```
>> airport(1).name
ans =
Reagan Washington
```

**Note:**  
 Children are referenced  
 by the `parent.child`  
 notation

- Check the contents of `airport(1).type`

```
>> airport(1).type
ans =
Public
```

# Inspecting the Contents of Struct Array Airport

- Check the contents of `airport(1).location`

```
>> airport(1).location
```

```
ans =
```

```
latitude: 38.8500  
longitude: 77.0333  
elevation: 15
```

The “child”  
called “location” has  
3 numeric pieces of  
data:

Latitude (degrees)  
Longitude (degrees)  
Elevation (feet)

# Usefulness of Struct Arrays

- Fast to retrieve since they are .mat files
- Organize data in a logical and convenient way
- Can contain dissimilar structures (i.e., numeric and string data)
- Can manipulate to do complex operations or to plot data (shown next)

# Manipulating Struct Array “Airport”

- Task: retrieve the locations of the airports contained in airport struct array and plot their locations in a map
- Plot : `airport(i).location.longitude` vs `airport(i).location.latitude`
- We use a simple map contained in a binary file called “usamap”
- Both files (`airport.mat`) and (`usamap.mat`) are available in the syllabus page



# Matlab Code

```

% Script to manipulate a struct file called airport

% Load the struct array first
load airport

% Check the length of the struct array using "length" command
% Assign the length of the struct file to a variable called
% "numberOfAirports"

numberOfAirports = length(airport);    % retrieves the length of the array

% Create a for loop to plot the locations of these airports

% Load USA map

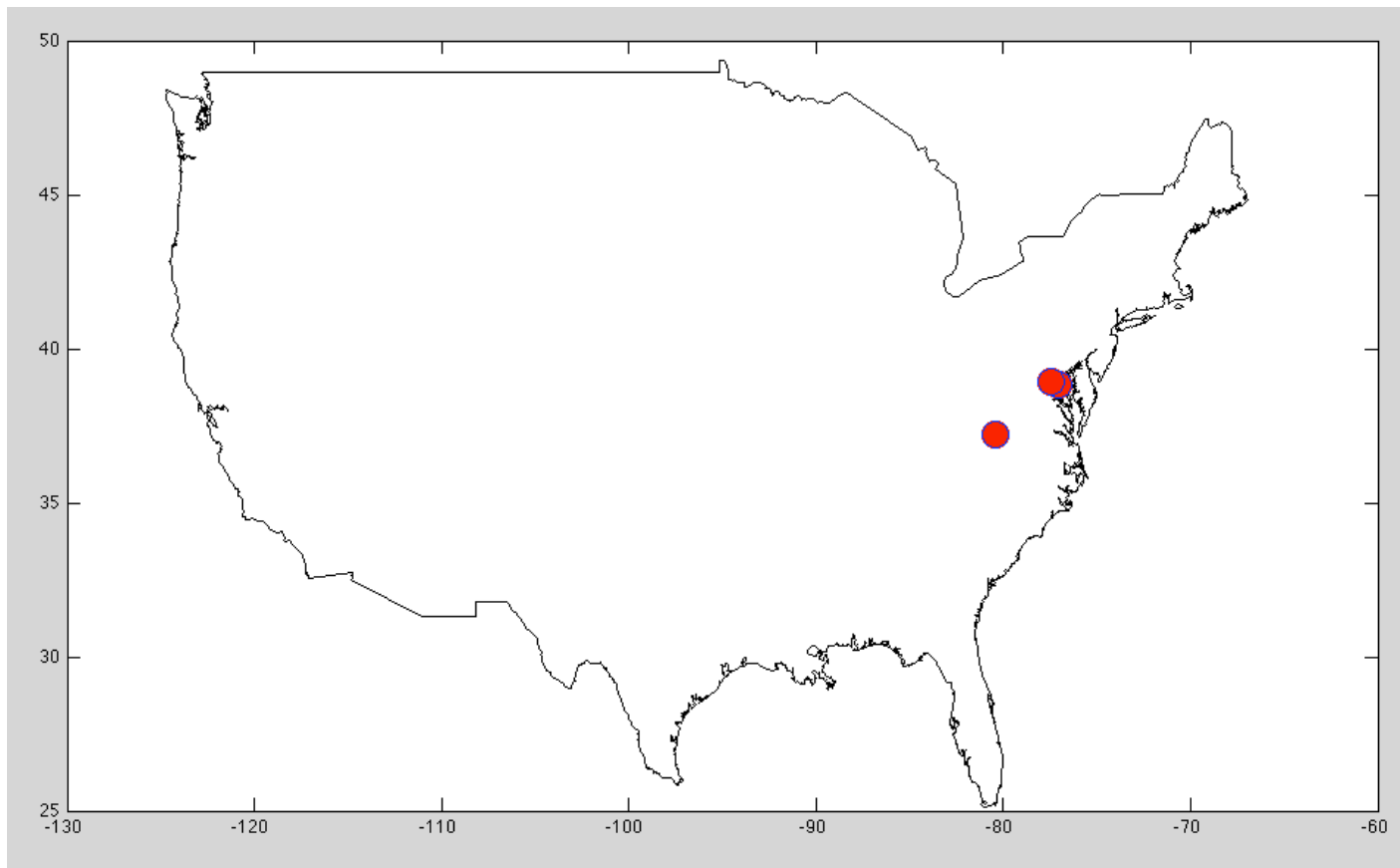
load usamap
    plot(uslon,uslat,'k')

hold on    % keeps the plot active throughout the for loop
for i=1:numberOfAirports
    plot(airport(i).location.longitude,airport(i).location.latitude,'o')
end

```

# Output of the Simple Matlab Code

- A simple map is generated after the script executes
- Note the locations of the airports



# To Do in Class

- Add x and y labels to the code
- Change the font size to make them better looking
- Try adding another airport to the data base:
  - `airport(4).name = 'San Francisco'`
  - `airport(4).type = 'Public'`
  - `airport(4).location.latitude = 37.62`
  - `airport(4).location.longitude = 122.38`
  - `airport(4).location.elevation = 13`

# Reading Excel Data Files with Matlab

- Using the **xlsread** Command
- Here is a sample script to read a data file containing data on bridges of the world

```
[num,txt,row] = xlsread
('bridges_of_the_world_short.xls','Bridge data');
```

- Reads the Excel worksheet named 'Bridge data' contained in file called **'bridges\_of\_the\_world\_short.xls'**
- Assigns all numeric data to variable '**num**'
- Assigns all text data to variable called '**txt**'
- All other unassigned data is stored in variable '**raw**'

# Review of Reading Data from Excel

- Now that you know how to define cell and struct arrays you can easily read data from Excel and store in any of the two learned Matlab structures
- The point is that many times is easier to do complex tasks in Matlab after you read an Excel file

# Excel File to be Read

	A	B	C	D
1	Name	Country	Completed	Length (m)
2	Mackinac	United States	1957	8038
3	Xiasha	China	1991	8230
4	Virginia-Dare-Memorial	United States	2002	8369
5	General-Rafael-Urdaneta	Venezuela	1962	8678
6	Sunshine-Skyway	United States	1987	8851
7	Twin-Span	United States	1960	8851
8	Wuhu-Yangtze-River	China	2000	10020
9	Third-Mainland	Nigeria	1991	10500
10	Seven-Mile	United States	1982	10887
11	San-Mateo-Hayward	United States	1967	11265
12	Leziria-Bridge	Portugal	2007	11670
13	Confederation	Canada	1997	12900
14	Rio-Niterol	Brazil	1974	13290
15	Kam-Sheung	Hong Kong	2003	13400
16	Penang	Malaysia	1985	13500
17	Vasco-da-Gama	Portugal	1998	17185
18	Bonnet-Carre-Spillway	United States	1960	17702
19	Chesapeake-Bay-Bridge-Tunnel	United States	1964	24140
20	Tianjin-Binhai	China	2003	25800
21	Atchafalaya-Swamp-Freeway	United States	1973	29290
22	Donghai	China	2005	32500
23	Manchac-Swamp	United States	1970	36710
24	Lake-Pontchartrain-Causeway	United States	1956	38422

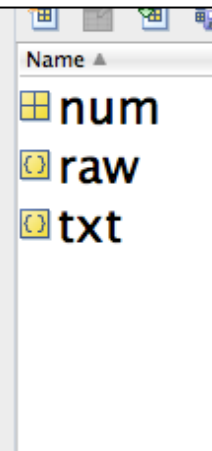
Bridges\_of\_the\_world\_short.xls

# What Happens after Executing the One Line Script?

- Three arrays are created using the previous script
- Array '**num**' is a standard matrix with size (23 x 2)
- Arrays '**raw**' and '**txt**' are cell arrays (24 x 4) each

```
>> whos
```

Name	Size	Bytes	Class	Attributes
num	23x2	368	double	
raw	24x4	12328	cell	
txt	24x4	11960	cell	



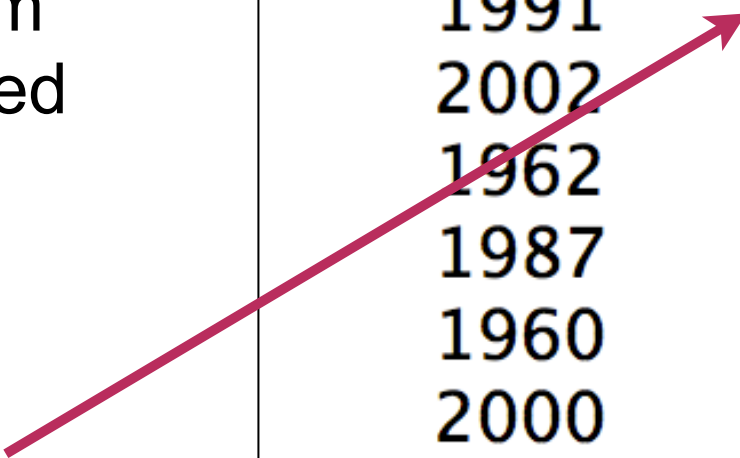
# Observations

- 'num' is a standard numeric array as shown
- Elements of 'num' can be referenced in the usual (row,column) format
- **num(2,2)=8230**

```
>> num

num =

    1957    8038
    1991    8230
    2002    8369
    1962    8678
    1987    8851
    1960    8851
    2000   10020
    1991   10500
    1982   10887
    1967   11265
```





# Observations (2)

- 'txt' is a cell array containing **string** data as shown
- Elements of 'txt' can be referenced using the cell array nomenclature `cell{i}(row,column)`
- **`txt{1,2}=Country`**

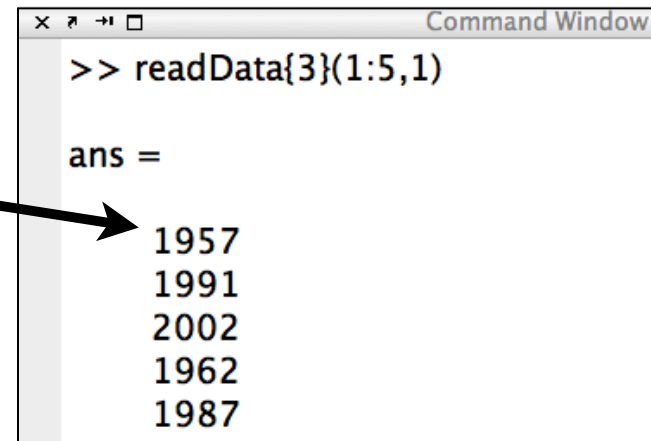
txt <24x4 cell>

	1	2	3	4
1	Name	Country	Completed	Length (m)
2	Mackinac	United States		
3	Xiasha	China		
4	Virginia-Da...	United States		
5	General-Raf...	Venezuela		
6	Sunshine-S.	United States		
7	Twin-Span	United States		
8	Wuhu-Yang...	China		
9	Third-Mainl...	Nigeria		
10	Seven-Mile	United States		
11	San Mateo-...	United States		
12	Leziria-Bridge	Portugal		
13	Confederation	Canada		
14	Rio-Niterol	Brazil		
15	Kam-Sheung	Hong Kong		
16	Penang	Malaysia		
17	Vasco-da-...	Portugal		
18	Bonnet-Car...	United States		
19	Chesapeake...	United States		
20	Tianjin-Binhai	China		
21	Atchafalaya...	United States		
22	Donghai	China		
23	Manchac-S...	United States		
24	Lake-Pontc...	United States		

# Note Differences in How Cell Arrays Store Information

- In previous case, a cell array storing numerical data can be referenced

- **readData{3}(1:5,1)**



```

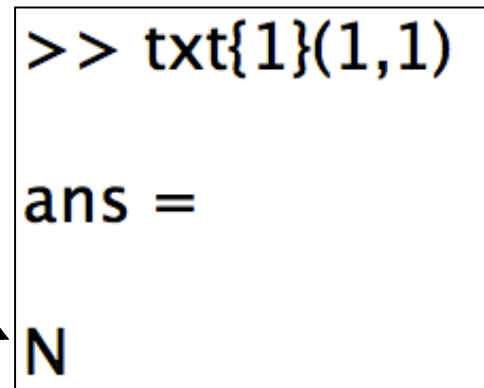
Command Window
>> readData{3}(1:5,1)

ans =

    1957
    1991
    2002
    1962
    1987
  
```

- In this last case, the cell array contains string information

- **txt{1}(1,2)=N**



```

>> txt{1}(1,2)

ans =

    N
  
```

# Matlab **xlsread** can Read a Range in an Excel

- The Matlab statement:
- `[num,txt,row] = xlsread('bridges_of_the_world_short.xls','Bridge data (A2:D24)');`
- Reads the Excel file but only across the range specified (A2:D24)
- This is useful if you know the data structure of the file you are reading