Previous Lecture:
- Characters and strings
- Intro to cell arrays

Today’s Lecture:
- More on cell arrays
- File input/output

Announcement:
- Project 4 due tonight at 11pm
Array vs. Cell Array

- **Simple array**
  - Each component stores **one scalar**. E.g., one char, one double, or one uint8 value
  - All components have the same type

- **Cell array**
  - Each cell can store something “bigger” than one scalar, e.g., a vector, a matrix, a string (vector of chars)
  - The cells may store items of different types
function D = CardDeck()
    % D is 1-by-52 cell array of strings that define a card deck

    suit = {'Hearts','Clubs','Spades','Diamonds'};
    rank = {'A','2','3','4','5','6','7','8','9','10','J','Q','K'};
    i = 1;  % index of next card
    for k= 1:4
        % Set up the cards in suit k
        for j= 1:13
            D{i}= [ rank{j} ' ' suit{k} ];
            i= i + 1;
        end
    end
Example: deal a 12-card deck

D:  

N: 1, 5, 9  
E: 2, 6, 10  
S: 3, 7, 11  
W: 4, 8, 12
% Deal a 52-card deck

N = cell(1,13); E = cell(1,13);
S = cell(1,13); W = cell(1,13);

for k=1:13
    N{k} = D{4*k-3};
    E{k} = D{4*k-2};
    S{k} = D{4*k-1};
    W{k} = D{4*k};
end

See function Deal
The “perfect shuffle” of a 12-card deck

A B C D E F G H I J K L
Perfect Shuffle, Step 1: cut the deck
Perfect Shuffle, Step 2: Alternate

A B C D E F G H I J K L

A B C D E F

G H I J K L

A G B H C I D J E K F L

1 2 3 4 5 6

1 2 3 4 5 6
Perfect Shuffle, Step 2: Alternate

A B C D E F G H I J K L

A B C D E F G H I J K L

A B C D E F

1 2 3 4 5 6

G H I J K L

2k

k

2 4 6 8 10 12

A G B H C I D J E K F L

Lecture 18
Perfect Shuffle, Step 2: Alternate

See function Shuffle
Application of cell array: sort data in a file

Suppose each line in the file

`statePop.txt`

is structured as follows:

Cols 1-14: State name
Cols 16-24: Population (millions)

The states appear in alphabetical order.
<table>
<thead>
<tr>
<th>State</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama</td>
<td>4557808</td>
</tr>
<tr>
<td>Alaska</td>
<td>663661</td>
</tr>
<tr>
<td>Arizona</td>
<td>5939292</td>
</tr>
<tr>
<td>Arkansas</td>
<td>2779154</td>
</tr>
<tr>
<td>California</td>
<td>36132147</td>
</tr>
<tr>
<td>Colorado</td>
<td>4665177</td>
</tr>
<tr>
<td>Texas</td>
<td>22859968</td>
</tr>
<tr>
<td>Utah</td>
<td>2469585</td>
</tr>
<tr>
<td>Vermont</td>
<td>623050</td>
</tr>
<tr>
<td>Virginia</td>
<td>7567465</td>
</tr>
<tr>
<td>Washington</td>
<td>6287759</td>
</tr>
<tr>
<td>West Virginia</td>
<td>1816856</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>5536201</td>
</tr>
<tr>
<td>Wyoming</td>
<td>509294</td>
</tr>
</tbody>
</table>
A detailed sort-a-file example

Create a new file

\texttt{statePopSm2Lg.txt}

that is structured the same as \texttt{statePop.txt} except that \textit{the states are ordered from smallest to largest according to population}.

\begin{itemize}
  \item Need the pop as \textit{numbers} for sorting.
  \item Can’t just sort the pop—have to maintain association with the state names.
\end{itemize}
First, read the file and store each line in a cell of a cell array

\[
C = \text{file2cellArray('StatePop')};
\]
In a file there are hidden “markers”

<table>
<thead>
<tr>
<th>State</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama</td>
<td>4557808</td>
</tr>
<tr>
<td>Alaska</td>
<td>663661</td>
</tr>
<tr>
<td>Arizona</td>
<td>5939292</td>
</tr>
</tbody>
</table>

geneData.txt

- Carriage return marks the end of a line
- `eof` marks the end of a file
Read data from a file

1. **Open** a file
2. **Read** it line-by-line until **end-of-file**
3. **Close** the file

**functions** `fopen`, `fgetl`, `feof`, `fclose`
1. Open the file

```matlab
fid = fopen('statePop.txt', 'r');
```

An open file has a file ID, here stored in variable `fid`.

Name of the file opened. `txt` and `dat` are common file name extensions for plain text files.

'`r`' indicates that the file has been opened for reading.

Built-in function to open a file.
2. Read each line and store it in cell array

```matlab
fid = fopen('statePop.txt', 'r');

k = 0;
while ~feof(fid)
    k = k + 1;
    Z{k} = fgetl(fid);
end
```

False until end-of-file is reached

Get the next line.
(Each call gets one line; you cannot make it skip lines or go to a specific line.)
3. Close the file

```matlab
fid = fopen('statePop.txt', 'r');

k = 0;
while ~feof(fid)
    k = k+1;
    Z{k} = fgetl(fid);
end

fclose(fid);
```
function CA = file2cellArray(fname)
% fname is a string that names a .txt file
% in the current directory.
% CA is a cell array with CA{k} being the
% k-th line in the file.

fid= fopen([fname '.txt'], 'r');
k= 0;
while ~feof(fid)
    k= k+1;
    CA{k}= fgetl(fid);
end
fclose(fid);
C

\{
\text{'Alab 4558000'}
\text{'Alas 664000'}
\vdots
\text{'Cali 36132000'}
\}

\{
\text{'Verm 623000'}
\vdots
\text{'Wyom 509000'}
\}

cell array
of strings
in alpha.order
C

{ 'Lab' : 4558000 ,
  'Lac' : 6640000 ,
  'Cali' : 36132000 ,
  ....
  'Verm' : 6230000 ,
  'Wyom' : 5090000 }

cell array of strings in alpha order

Crew

{ 'Wyom' : 5090000 ,
  'Verm' : 6230000 ,
  ....
  'Cali' : 36132000 }
C

\[
\begin{array}{l}
\text{'Alab 4558000'} \\
\text{'Alas 664000'} \\
\vdots \\
\text{'Cali 36132000'} \\
\text{'Verm 623000'} \\
\vdots \\
\text{'Wyom 509000'}
\end{array}
\]

cell array of strings in alpha-order

Pop

\[
\begin{bmatrix}
4558000 \\
664000 \\
\vdots \\
36132000 \\
623000 \\
\vdots \\
509000
\end{bmatrix}
\]

vector of numbers

Crew

\[
\begin{array}{l}
\text{'Wyom 509000'} \\
\text{'Verm 623000'} \\
\vdots \\
\text{'Cali 36132000'}
\end{array}
\]
Next, get the populations into a numeric vector

\[
C = \text{file2cellArray}('\text{StatePop}');
\]
\[
n = \text{length}(C);
\]
\[
\text{pop} = \text{zeros}(n,1);
\]
\[
\text{for } i=1:n
\]
\[
\quad S = C\{i\};
\]
\[
\quad \text{pop}(i) = \text{str2double}(S(16:24));
\]
\[
\text{end}
\]

Converts a \textit{string} representing a numeric value (digits, decimal point, spaces) to the numeric value \(\rightarrow\) scalar of type \textit{double}. E.g., \(x=\text{str2double(''-3.24'')}\) assigns to variable \(x\) the numeric value -3.2400...
C

\{
\{ 'Alab 4558000' \}
\{ 'Alas 664000' \}
\{ 'Cali 36132000' \}
\{ 'Verm 623000' \}
\{ 'Wyom 509000' \}
\}

\begin{bmatrix}
4558000 \\
664000 \\
36132000 \\
623000 \\
509000 \\
\end{bmatrix}

\begin{bmatrix}
4558000 \\
664000 \\
36132000 \\
623000 \\
509000 \\
\end{bmatrix}

\{
\{ 'Wyom 509000' \}
\{ 'Verm 623000' \}
\{ 'Cali 36132000' \}
\}

cell array of strings in alpha order

vector of numbers
C
\[
\{ 'Alab', 4558000 \\
'Alas', 6640000 \\
'\vdots' \\
'Cali', 36132000 \\
'Verm', 6230000 \\
'Wyom', 5090000
\}
\]

Pop
\[
\left[ \begin{array}{c}
4558000 \\
6640000 \\
36132000 \\
6230000 \\
5090000
\end{array} \right]
\]

s
\[
\left[ \begin{array}{c}
5090000 \\
6230000 \\
5090000 \\
36132000 \\
50
\end{array} \right]
\]

idx
\[
\left[ \begin{array}{c}
45 \\
45 \\
36132000 \\
5
\end{array} \right]
\]

Gnew
\[
\{ 'Wyom', 5090000 \\
'Verm', 6230000 \\
'\vdots' \\
'Cali', 36132000
\}
\]

Cell array of strings in alpha. order

Vector of numbers

Vector of indices (ranks)
Built-In function `sort`

Syntax: \[ y, \text{idx} \] = sort(x) 

\[
\begin{array}{c}
\text{x:} \\
10 & 20 & 5 & 90 & 15 \\
\end{array}
\]

\[
\begin{array}{c}
\text{y:} \\
5 & 10 & 15 & 20 & 90 \\
\end{array}
\]

\[
\begin{array}{c}
\text{idx:} \\
3 & 1 & 5 & 2 & 4 \\
\end{array}
\]

\[ y(1) = x(3) = x(\text{idx}(1)) \]
Built-In function `sort`  

Syntax:  \[ y, \text{idx} \] = sort(x)  

<table>
<thead>
<tr>
<th>x:</th>
<th>10</th>
<th>20</th>
<th>5</th>
<th>90</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>y:</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>90</td>
</tr>
<tr>
<td>idx:</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

\[ y(2) = x(1) = x(\text{idx}(2)) \]
Built-In function **sort**

Syntax: \[ y, \text{idx} \] = \text{sort}(x) 

\[
\begin{align*}
x: & & 10 & 20 & 5 & 90 & 15 \\
y: & & 5 & 10 & 15 & 20 & 90 \\
\text{idx}: & & 3 & 1 & 5 & 2 & 4
\end{align*}
\]

\[ y(3) = x(5) = x(\text{idx}(3)) \]
Built-In function \texttt{sort}

Syntax: \[ y, idx \] = sort(x)
C

\(\begin{align*}
\text{\{'Alab} & \text{ 4558000'} \\
\text{\{'Alas} & \text{ 664000'} \\
\vdots & \phantom{\text{\{'Alas}} \\
\text{\{'Cali} & \text{ 36132000'} \\
\text{\{'Verm} & \text{ 623000'} \\
\vdots & \phantom{\text{\{'Verm}} \\
\text{\{'Wyom} & \text{ 509000'}
\end{align*}\}
\)

\text{cell array of strings in alpha. order}

\begin{align*}
\text{Pop} & \phantom{\text{\{'Alab}} \\
\begin{bmatrix}
4558000 \\
664000 \\
36132000
\end{bmatrix}
\end{align*}

\begin{align*}
\text{s} & \phantom{\text{\{'Alab}} \\
\begin{bmatrix}
509000 \\
623000 \\
36132000
\end{bmatrix}
\end{align*}

\begin{align*}
\text{idx} & \phantom{\text{\{'Alab}} \\
\begin{bmatrix}
50 \\
45 \\
5
\end{bmatrix}
\end{align*}

\text{Cnew}

\(\begin{align*}
\text{\{'Wyom} & \text{ 509000'} \\
\text{\{'Verm} & \text{ 623000'} \\
\vdots & \phantom{\text{\{'Verm}} \\
\text{\{'Cali} & \text{ 36132000'}
\end{align*}\}
\)

\text{vector of indices (ranks)}

\text{vector of numbers}
Sort from little to big

% C is cell array read from statePop.txt
% pop is vector of state pop (numbers)
[s, idx] = sort(pop);
Cnew = cell(n,1);
for i=1:length(Cnew)
    ithSmallest = idx(i);
    Cnew{i} = C{ithSmallest};
end
<table>
<thead>
<tr>
<th>State</th>
<th>Population</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wyoming</td>
<td>509294</td>
</tr>
<tr>
<td>Vermont</td>
<td>623050</td>
</tr>
<tr>
<td>North Dakota</td>
<td>636677</td>
</tr>
<tr>
<td>Alaska</td>
<td>663661</td>
</tr>
<tr>
<td>South Dakota</td>
<td>775933</td>
</tr>
<tr>
<td>Delaware</td>
<td>843524</td>
</tr>
<tr>
<td>Montana</td>
<td>935670</td>
</tr>
<tr>
<td>Illinois</td>
<td>12763371</td>
</tr>
<tr>
<td>Florida</td>
<td>17789864</td>
</tr>
<tr>
<td>New York</td>
<td>19254630</td>
</tr>
<tr>
<td>Texas</td>
<td>22859968</td>
</tr>
<tr>
<td>California</td>
<td>36132147</td>
</tr>
</tbody>
</table>
Sort from little to big

% C is cell array read from statePop.txt
% pop is vector of state pop (numbers)
[s,idx] = sort(pop);
Cnew = cell(n,1);
for i=1:length(Cnew)
    ithSmallest = idx(i);
    Cnew{i} = C{ithSmallest};
end

cellArray2file(Cnew,'statePopSm2Lg')
A 3-step process to read data from a file or write data to a file

1. **(Create and ) open** a file
2. **Read** data from or **write** data to the file
3. **Close** the file
Open a file

\[ \text{fid} = \text{fopen}(\text{'popSm2Lg.txt'}, \text{'w'}); \]

- An open file has a file ID, here stored in variable `fid`.
- Name of the file (created and) opened. `txt` and `dat` are common file name extensions for plain text files.
- `'w'` indicates that the file is to be opened for writing.
- Use `'a'` for appending.

Built-in function to open a file.
2. Write (print) to the file

```matlab
fid = fopen('popSm2Lg.txt', 'w');

for i=1:length(Z)
    fprintf(fid, '%s
', Z{i});
end
```

Substitution sequence specifies the string format (followed by a new-line character)

The $i^{th}$ item in cell array $Z$
2. Write (print) to the file

```matlab
fid = fopen('popSm2Lg.txt', 'w');
for i=1:length(Z)
    fprintf(fid, '%s\n', Z{i});
end
```

- Printing is to be done to the file with ID `fid`
- Substitution sequence specifies the string format (followed by a new-line character)
- The $i^{th}$ item in cell array `Z`
3. Close the file

```matlab
fid = fopen('popSm2Lg.txt', 'w');

for i=1:length(Z)
    fprintf(fid, '%s\n', Z{i});
end

fclose(fid);
```

```matlab
fclose(fid);
```
function cellArray2file(CA, fname)
% CA is a cell array of strings.
% Create a .txt file with the name
% specified by the string fname.
% The i-th line in the file is CA{i}

fid= fopen([fname `.txt' ], 'w');
for i= 1:length(CA)
    fprintf(fid, '%s\n', CA{i});
end
fclose(fid);
Storing only a selected (small) section of data from a big file

- The previous example reads the whole file and stores all the text
- If you’re interested in only a small part of the data, storing everything is an overkill
- Read “Extra file i/o example” posted on the website to learn how to store only the data that meet certain criteria
I want to put in the 3rd cell of cell array C a single string. Which is correct?

A. \( C\{3\} = \text{‘a cat’}; \)
B. \( C\{3\} = [\text{‘a } \text{‘cat’}]; \)
C. \( C(3) = \{\text{‘a } \text{‘cat’}\}; \)
D. Two answers above are correct
E. Answers A, B, C are all correct