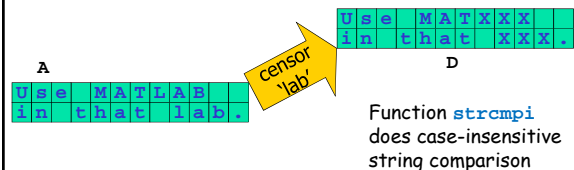


- Previous Lecture:
 - Characters and strings
- Today's Lecture:
 - More on characters and strings
 - Cell arrays
 - File input/output
- Announcements:
 - Discussion this week in computer lab
 - Project 4 due on Wednesday at 11pm

Example: censoring words

```
function D = censor(str, A)
% Replace all occurrences of string str in
% character matrix A with X's, regardless of
% case.
% Assume str is never split across two lines.
% D is A with X's replacing str.
```



Lecture 18

6

```
function D = censor(str, A)
% Replace all occurrences of string str in character matrix A,
% regardless of case, with X's.
% A is a matrix of characters.
% str is a string. Assume that str is never split across two lines.
% D is A with X's replacing the censored string str.

D = A;
ns = length(str);
[nr,nc] = size(A);

% Build a string of X's of the right length

% Traverse the matrix to censor string str
```

Lecture 17

7

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

Lecture 19

10

1-d and 2-d examples ...

Vectors and matrices store values of the same type in all components

5 x 1 matrix

4 x 5 matrix

Cell array: individual components may contain different types of data

3 x 2 cell array

Lecture 18

11

Cell Arrays of Strings

```
C = { 'Alabama', 'New York', 'Utah' }
```

```
C = { 'Alabama'; 'New York'; 'Utah' }
```

1-d cell array of strings

Contrast with
2-d array of characters

```
M = [ 'Alabama'; ...
      'New York'; ...
      'Utah' ]
```

Use braces {} for creating and addressing cell arrays

Matrix	Cell Array
<ul style="list-style-type: none"> Create <pre>m = [5, 4; ... 1, 2; ... 0, 8]</pre>	<ul style="list-style-type: none"> Create <pre>C = { ones(2,2), 4; ... 'abc', ones(3,1); ... 9, 'a cell' }</pre>
<ul style="list-style-type: none"> Addressing <pre>m(2,1) = pi</pre>	<ul style="list-style-type: none"> Addressing <pre>C{2,1} = 'ABC' C{3,2} = pi disp(C{3,2})</pre>

Lecture 18

13

Creating cell arrays...

```
C = {'Oct', 30, ones(3,2)};
is the same as
C = cell(1,3); % not necessary
C{1} = 'Oct';
C{2} = 30;
C{3} = ones(3,2);
```

You can assign the empty cell array: `D = {}`

Lecture 18

14

Example: Represent a deck of cards with a cell array

```
D{1} = 'A Hearts';
D{2} = '2 Hearts';
:
D{13} = 'K Hearts';
D{14} = 'A Clubs';
:
D{52} = 'K Diamonds';
```

But we don't want to have to type all combinations of suits and ranks in creating the deck... How to proceed?

Lecture 18

15

Make use of a suit array and a rank array ...

```
suit = {'Hearts', 'Clubs', ...
        'Spades', 'Diamonds'};
rank = {'A','2','3','4','5','6',...
        '7','8','9','10','J','Q','K'};
```

Then concatenate to get a card. E.g.,

```
str = [rank{3} ' ' suit{2} ];
D{16} = str;
```

So `D{16}` stores '3 Clubs'

Lecture 18

16

To get all combinations, use nested loops

```
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
```


See function `CardDeck`


Lecture 18


17


Example: deal a 12-card deck

D: 

N:  1,5,9 4k-3

E:  2,6,10 4k-2

S:  3,7,11 4k-1

W:  4,8,12 4k

Lecture 18

19

```
% Deal a 52-card deck
N = cell(1,13); E = cell(1,13);
S = cell(1,13); W = cell(1,13);

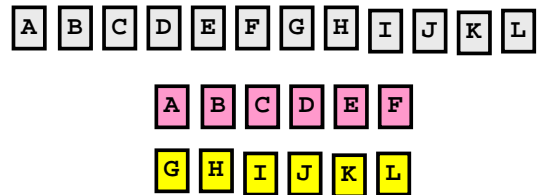
for k=1:13
    N{k} = D{4*k-3};
    E{k} = D{4*k-2};
    S{k} = D{4*k-1};
    W{k} = D{4*k};
end
```

See function Deal

Lecture 18

20

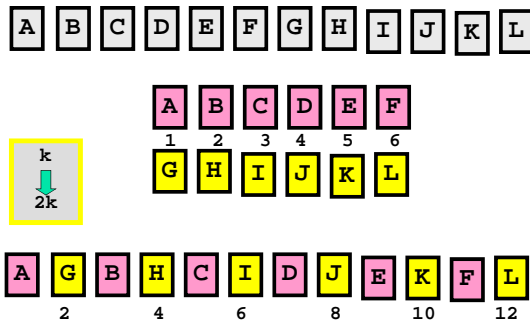
Perfect Shuffle, Step 1: cut the deck



Lecture 18

22

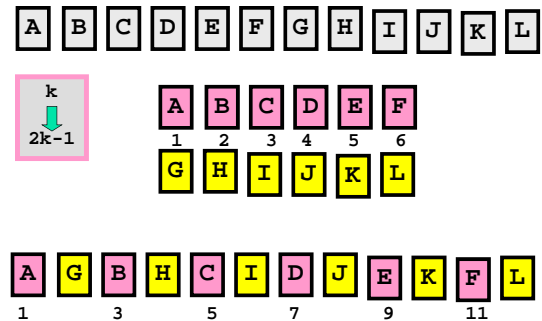
Perfect Shuffle, Step 2: Alternate



Lecture 18

24

Perfect Shuffle, Step 2: Alternate



Lecture 18

25

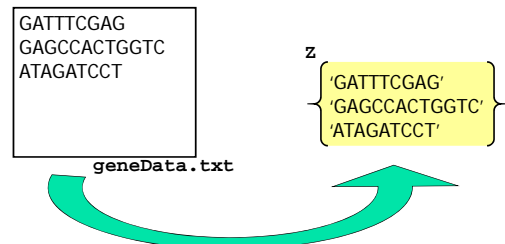
A 3-step process to
read data from a file or
write data to a file

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

Lecture 18

31

Working with data files: Read the data in a file
line-by-line and store the results in a cell array



How are lines separated?
How do we know when there are no more lines?

Lecture 18

32

In a file there are hidden “markers”

```
GATTCGAG •
GAGCCACTGGTC •
ATAGATCCT •
■
```

geneData.txt

- Carriage return marks the end of a line

■ eof marks the end of a file

Lecture 18

33

1. Open the file

```
fid = fopen('geneData.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

Built-in function to open a file

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

Lecture 18

35

2. Read each line and store it in cell array

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

```
k= 0;
```

```
while ~feof(fid)
```

False until end-of-file is reached

```
    k= k+1;
```

```
    Z{k}= fgetl(fid);
```

```
end
```

Get the next line

Lecture 18

36

3. Close the file

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

```
k= 0;
```

```
while ~feof(fid)
```

```
    k= k+1;
```

```
    Z{k}= fgetl(fid);
```

```
end
```

```
fclose(fid);
```

Lecture 18

37

```
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);
```

Lecture 18

38

Example: subset of clicker IDs

```
IDs
```

```
['d091314'; ...
 'h134d83'; ...
 'h4567s2'; ...
 'fr83209']
```

Find subset that begins with 'h'

```
L
```

```
{'h134d83', ...
 'h4567s2'}
```

```
L= {};
```

```
k= 0;
```

```
for r=1:size(IDs,1)
```

```
    if IDs(r,1)=='h'
```

```
        k= k+1;
```

```
        L{k} = IDs(r,:);
```

```
    end
```

```
end
```

Directly assign into a particular cell—good!

```
L= {};
```

```
for r=1:size(ID,1)
```

```
    if IDs(r,1)=='h'
```

```
        L= [L, IDs(r,:)];
```

```
    end
```

```
end
```

Concatenate cells or cell arrays—prone to problems!