- Today's Lecture:
  - 1-d and 2-d arrays of type `char`
  - Computing with characters

Su	M	Tu	W	Th	F	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

- Announcements:
  - A1 resubmissions currently being graded
  - Assignment 2 first submission due Monday 9/27
  - Mon lecture: Review session for Test 1
  - Test on Wed 9/29 in Thurston 205, 2:40-3:30pm

Character array (an array of type `char`)

- We have used strings of characters in programs already:
  - `n= input('Next number: ')`
  - `sprintf('Answer is %f', ans)`
- A string is made up of individual characters, so a string is a 1-d array of characters
- `'CS1132 rocks!'` is a character array of length 13; it has 7 letters, 4 digits, 1 space, and 1 symbol.

'C'	'S'	'1'	'1'	'3'	'2'	' '	'r'	'o'	'c'	'k'	's'	'!'

Row vector of length 13

- Can have 2-d array of characters as well

'C'	'S'	'1'	'1'	'3'	'2'
'r'	'o'	'c'	'k'	's'	'!'

2×6 matrix

Recap: Single quotes enclose char arrays in Matlab

Anything enclosed in single quotes is a string (*even if it looks like something else*)

- `'100'` is a character array (string) of length 3
- `100` is a numeric value
- `'pi'` is a character array of length 2
- `pi` is the built-in constant 3.14159…
- `'x'` is a character (vector of length 1)
- `x` may be a variable name in your program

Types so far: `char`, `double`, `logical`

```
a= 'CS1'
a= ['C','S','1']
```

a is a 1-d array with type `char` components. Often called a *string*; NOT the same as a *new* type in Matlab 2017+ called `string`.

a

'C'	'S'	'1'

```
b= [3 9]
```

b is a 1-d array with type `double` components. `double` is the default type for numbers in Matlab. We call b a “numeric array”

```
d= rand() > .5
```

d is a scalar of the type `logical`. We call d a “Boolean value”

Basic (simple) types in MATLAB

- E.g., `char`, `double`, `logical`
- Each uses a set amount of memory
  - Each `double` value uses 64 bits (=8 bytes)
  - Each `char` value uses 16 bits (=2 bytes)
  - Use function `whos` to see memory usage by variables in workspace
- Can easily determine amount of memory used by a simple array (array of a basic type, where each component stores one simple value)
- Later: Special arrays where each component is a container for a collection of values

Text—sequences of characters often called strings—are important in computation

Numerical data is often encoded in strings. E.g., a file containing Ithaca weather data begins with the string

`W07629N4226`

meaning

Longitude:	76º 29′ West
Latitude:	42º 26′ North

We may need to grab hold of the substring `W07629`, convert `076` and `29` to the numeric values 76 and 29, and do some computation

A text sequence is a vector (of characters)

Vectors

- Assignment

```
v= [7, 0, 5];
```

- Indexing

```
x= v(3);      % x is 5
v(1)= 1;      % v is [1 0 5]
w= v(2:3);    % w is [0 5]
```

- : notation

```
v= 2:5;       % v is [2 3 4 5]
```

- Appending

```
v= [7 0 5];
v(4)= 2;      % v is [7 0 5 2]
```

- Concatenation

```
v= [v [4 6]];
      % v is [7 0 5 2 4 6]
```

Strings

- Assignment

```
s= ['h','e','l','l','o'];
                % formal
s= 'hello';     % shortcut
```

- Indexing

```
c= s(2);      % c is 'e'
s(1)= 'J';    % s is 'Jello'
t= s(2:4);    % t is 'ell'
```

- : notation

```
s= 'a':'g'; % s is 'abcdefg'
```

- Appending

```
s= 'duck';
s(5)= 's';    % s is 'ducks'
```

- Concatenation

```
s= [s ' quack'];
         % s is 'ducks quack'
```

Example: removing all occurrences of a character

- From a genome bank we get a sequence

  `ATTG CCG TA  GCTA CGTACGC AACTGG AAATGGC CGTAT…`

- First step is to “clean it up” by removing all the blanks. Write this function:

```
function s = removeChar(c, s)
% Return char array s with all occurrences of
% char scalar c removed.
```

Example: removing all occurrences of a character

- Can solve this problem using iteration—check one character (one component of the vector) at a time
- Challenge: Can you solve it using logical indexing?

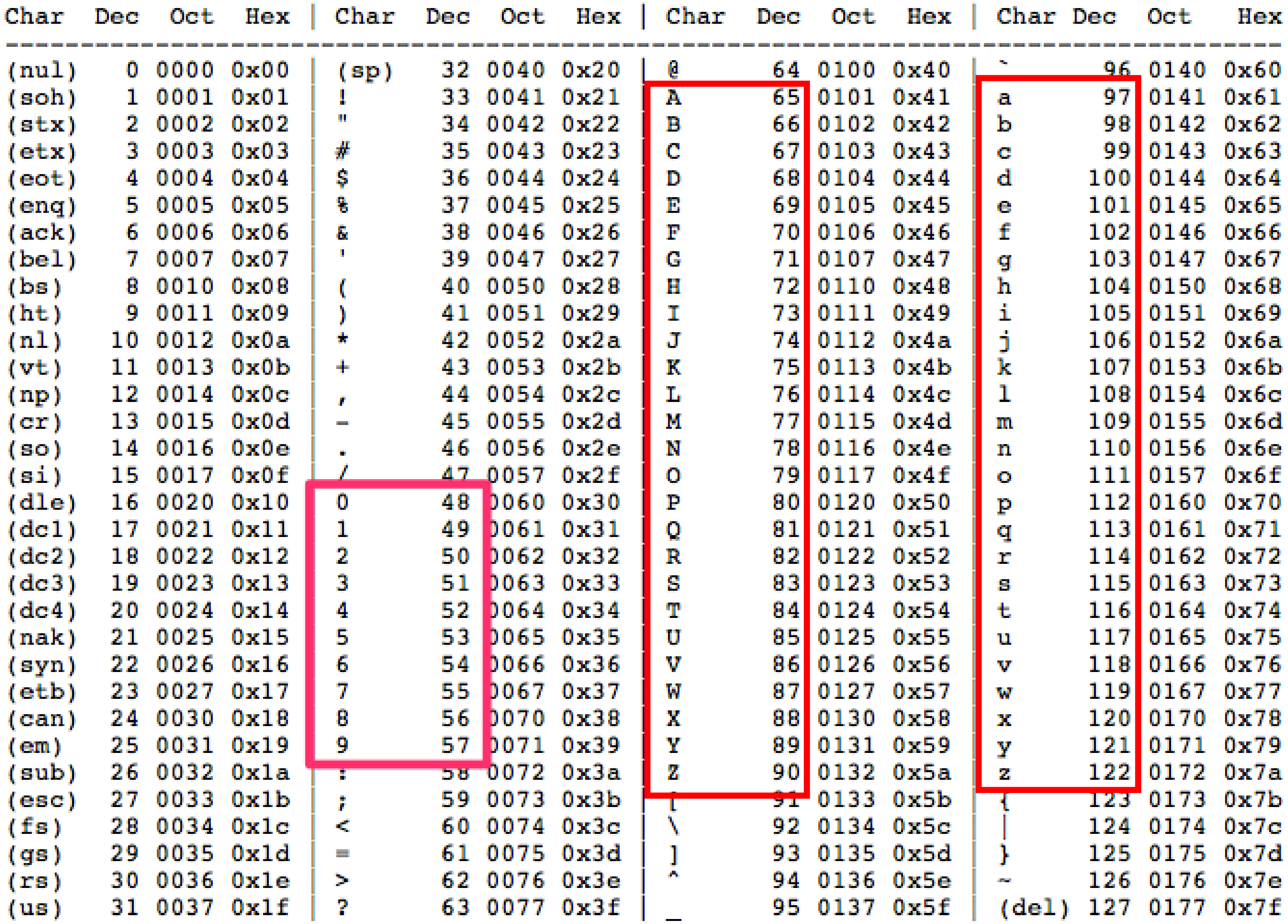
t= s(s ~= c)

```
function t = removeChar_loop(c, s)
% Return char array s with all
% occurrences of char scalar c
% removed.
t= '';
for k = 1:length(s)
    if s(k) ~= c
        t= [t s(k)];
    end
end
```

Some useful char array functions

```
s= 'Matlab 1132';

length(s)   % 11
isletter(s) % [1 1 1 1 1 1 0 0 0 0 0]
isspace(s)  % [0 0 0 0 0 0 1 0 0 0 0]
lower(s)    % ‘matlab 1132'
upper(s)    % ‘MATLAB 1112'

ischar(s)
  % Is s a char array? True (1)
strcmp(s(1:3), 'mat')
  % Compare strings str(1:3) & 'mat'. False (0)
strcmp(s(1:3), 'Ma')
  % False (0)
```

The ASCII Table



Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f

ASCII characters

(American Standard Code for Information Interchange)

ascii code	*Character*
:	:
:	:
65	‘A’
66	‘B’
67	‘C’
:	:
90	‘Z’
:	:

ascii code	*Character*
:	:
:	:
48	‘0’
49	‘1’
50	‘2’
:	:
57	‘9’
:	:

Character vs Unicode code points

```
str= ’Age 19’
    %a 1-d array of characters
code= double(str)
    %convert chars to Unicode values
str1= char(code)
    %convert Unicode values to chars
```

Arithmetic and relational ops on characters

- `'c'-'a'` gives 2
- `'6'-'5'` gives 1
- `letter1='e'; letter2='f';`
- `letter1-letter2` gives -1

- `'c'>'a'` gives true
- `letter1==letter2` gives false

- `'A' + 2` gives 67
- `char('A'+2)` gives ‘C’

What is in variable g (if it gets created)?

```
d1= 'Mar 3';  d2= 'Mar 9';
x1= d1(5);    x2= d2(5);
g= x2-x1;
```

Alfa: the character ‘6’

Bravo: the numeric value 6

Charlie: Error in assigning variables **x1**, **x2**

Delta: Error in the subtraction operation

Echo: Some other value or error

What is in variable **g** (if it gets created)?

```
d1= 'Mar 13';   d2= 'Mar 29';
x1= d1(5:6);    x2= d2(5:6);
g= x2-x1;
```

Alfa: the string ‘**16**’

Bravo: the numeric value 16

Charlie: Error in assigning variables **x1**, **x2**

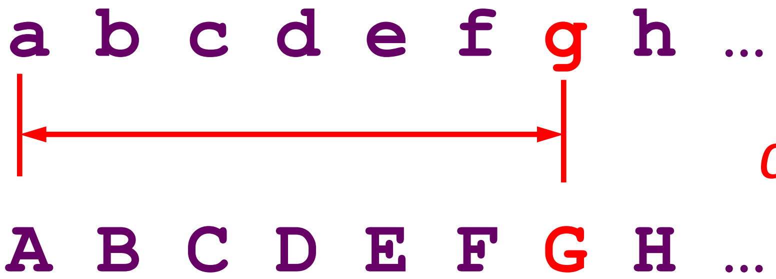
Delta: Error in the subtraction operation

Echo: Some other value or error

Example: toUpper

Write a function toUpper(cha) to convert character cha to upper case if cha is a lower case letter. Return the converted letter. If cha is not a lower case letter, simply return the character cha.

Hint: Think about the distance between a letter and the base letter 'a' (or 'A'). E.g.,



a b c d e f g h …

A B C D E F G H …

distance = 'g'-'a' = 6 = 'G'-'A'

Of course, do not use Matlab function upper!

```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.
```

```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;
```

cha is lower case if it is between 'a' and 'z'

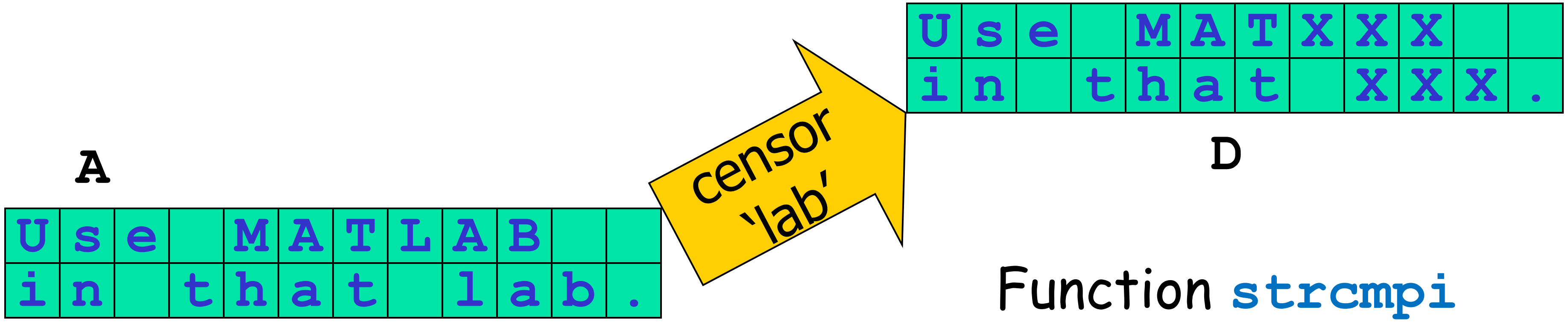
```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from ‘a’




end
```

```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from ‘a’
    offset= cha - 'a';

    % Go same distance from ‘A’

end
```

```
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from ‘a’
    offset= cha - 'a';

    % Go same distance from ‘A’
    up= char('A' + offset);
end
```

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.
```



A

U	s	e		M	A	T	L	A	B		
i	n		t	h	a	t		l	a	b	.

U	s	e		M	A	T	X	X	X		
i	n		t	h	a	t		X	X	X	.

D

Function **strcmpi** does case-insensitive string comparison

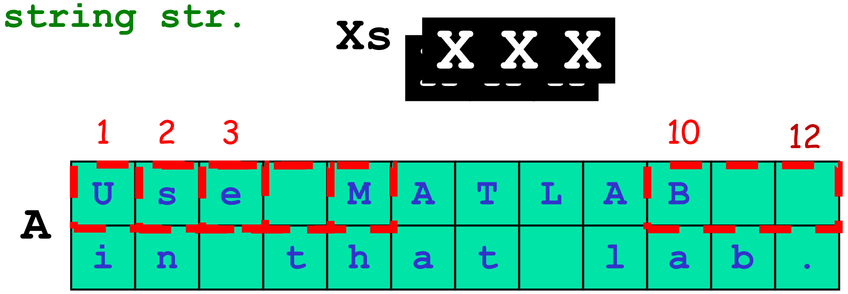
```
function D = censor(str, A)
% Replace all occurrences of string str in character matrix A,
% regardless of case, with X's.
% A is a matrix of characters.
% str is a string.  Assume that str is never split across two lines.
% D is A with X's replacing the censored string str.

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

A

	1	2	3		... c ...			8	9	10	11	12
1	U	s	e		M	A	T	L	A	B		
2	i	n		t	h	a	t		l	a	b	.

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





% Traverse the matrix to censor string str
```

```
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
Xs= char( zeros(1,ns));
for k= 1:ns
    Xs(k)= 'X';
end

% Traverse the matrix to censor string str
for r= 1:nr
    for c= 1:nc-ns+1
        if  strcmpi( str , A(r, c:c+ns-1) )
            D(r, c:c+ns-1)= Xs;
        end
    end
end
```



U	s	e		M	A	T	L	A	B		
i	n		t	h	a	t		l	a	b	.

Returns an array of type **double**

Changes the type to **char**

Case insensitive comparison of strings