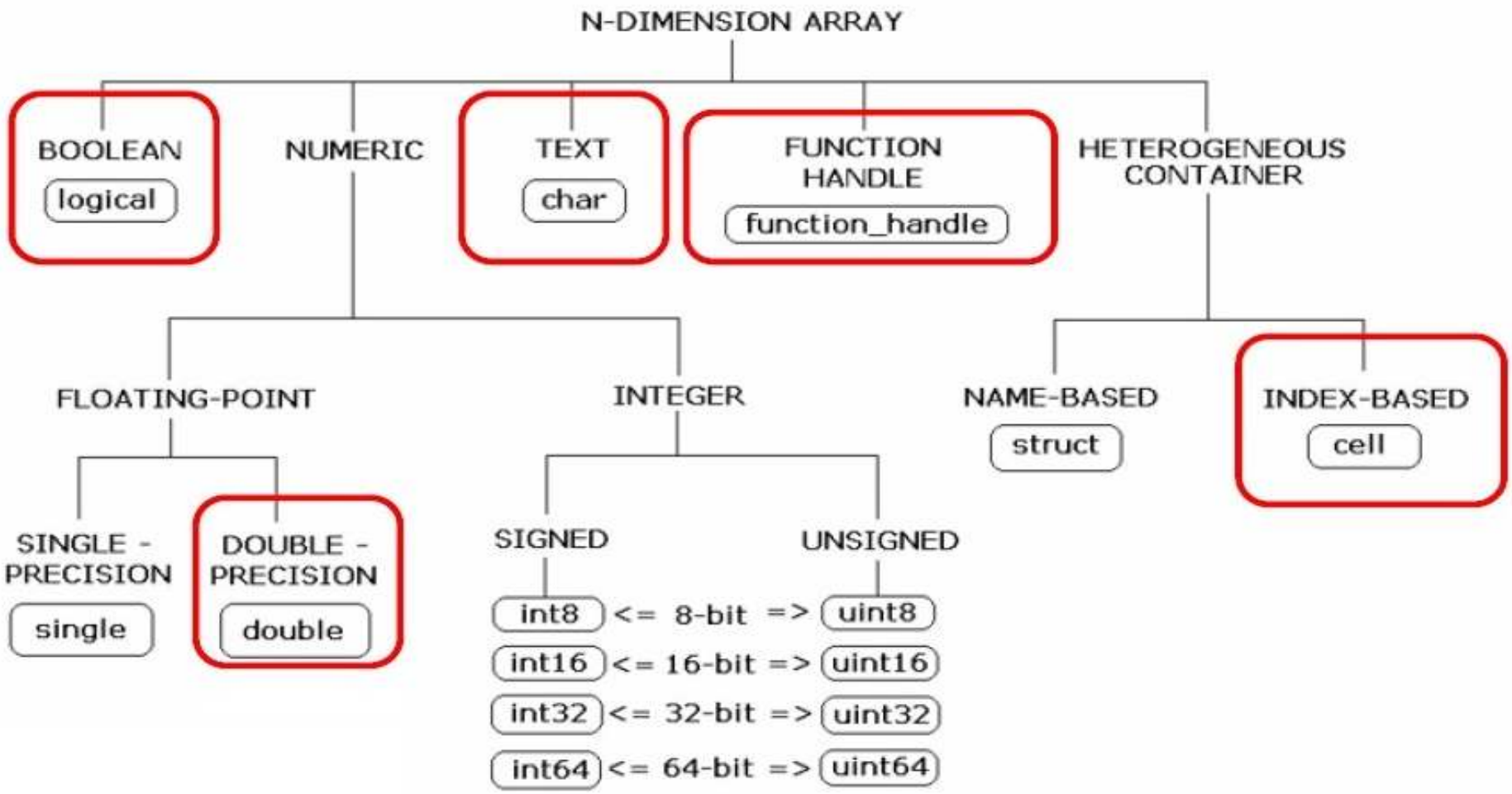
EEE 208 – Programming for EEE

Assist. Prof. Dr. Engin Mendi

Types/Classes of Variable in MATLAB

Variables have 3 features (attributes): class, size, value



Doubles and Characters

- All of the real and complex variables and vectors that we studied so far were of type (class) double.
- Each variable of type double has a space of up to 64 bits in the memory.
- But there are characters and strings, known as char
- Character arrays only use 2 bytes/element.
- The figure below shows the name, value, size, byte, min and max, and class for each variable in the workspace.

g	[0.9950 + 0.0998i,...	1x10	160	0.9950...	-0.32...	double (complex)
h	[0.0998 + 0.9950i,...	1x10	160	0.0998...	0.295...	double (complex)
k	65	1x1	8	65	65	double
m	[22,4,-7]	1x3	24	-7	22	double
mystring	'Hello World'	1x11	22			char
n	101	1x1	8	101	101	double
t	<1x161 double>	1x161	1288	1850	2010	double

Intro to Characters and Strings

- In most programming languages, the first text that students learn to write on the screen is Hello World!

- In MATLAB, we use the function `disp` to display strings between two single quotation marks. It displays the array, without printing the array's name:
- `>> disp('Hello World!')`

- You can use this function in the beginning of your code to explain what it does, or tell the user what variables he/she has to enter. You can also use it to display values of variables on the screen.

Characters and Strings, 1

- You can define string variables and use them in other functions, but don't forget to use quotation marks:

```
char1='I'
size(char1)
char2=' ' % the space character
size(char2)
char3='am'
size(char3)
```

We can use concatenation techniques:

- `string1 = [char1 char2 char3]`
- `size(string1)`
- Look at the workspace: string1 is also of the type char. What is its size?

Characters and Strings, 2

- Use integer indices to access specific elements of a char array.

- Remember how we tried to access the elements in vectors and matrices? We can do the same thing for a variable of type character.
- Example
- `>> sentence = 'Summer is the warmest season'`
- `>> size(sentence)`
- `>> sentence(1:4)`
- `>> sentence(end:-1:1)`
- `>> sentence([5 8 12])`

disp and datestr Functions

- `string1 = [char1 char2 char3]`

Each character is counted as 1 element.

```
>> disp(string1)

>> mystring = 'Hello World' ;
>> disp(mystring)
>> class(mystring)
>> size(mystring)
```

- Why? Because it has 1 row and 11 elements (characters)

Clock and datestr

- `>> help clock`

`C = clock` returns a six element date vector containing the current time and date in decimal form: [year month day hour minute seconds]

-

- `>> help datestr`

`datestr` converts date and time to string format

```
>> C = [2013, 3, 12, 11, 6, 23];
>> datestr(C)
```

Disp: explanation

1. When using `disp` to display a simple sentence or word, use parentheses and single quotation marks: `disp(' ')`
2. The text inside quotations becomes purple to show it has a correct format.
3. But if you want to show constant text and variables at the same time, you should parentheses and brackets: `disp([ ])`
4. To use any variable inside the `disp` function, you have to change it to a string or character type. For example, `datestr` converts vectors of clock output to a string.

Example: Clock

- We want to read the current clock, find its size, then convert it to a string using a specific format, and display the data on the screen using disp function.

```
>> start = clock
>> size(start)
>> CurrentClock = datestr(start)
>> disp(['The current date and time are '
CurrentClock])
>> disp(['The current date is ' CurrentClock(1:11)
' and the current time is ' CurrentClock(13:end)])
```

- Don't forget the brackets [ ]

Flow Control: IF

IF

```
if cond
    commands
end
```

Conditional statement: evaluates to true or false

ELSE

```
if cond
    commands1
else
    commands2
end
```

ELSEIF

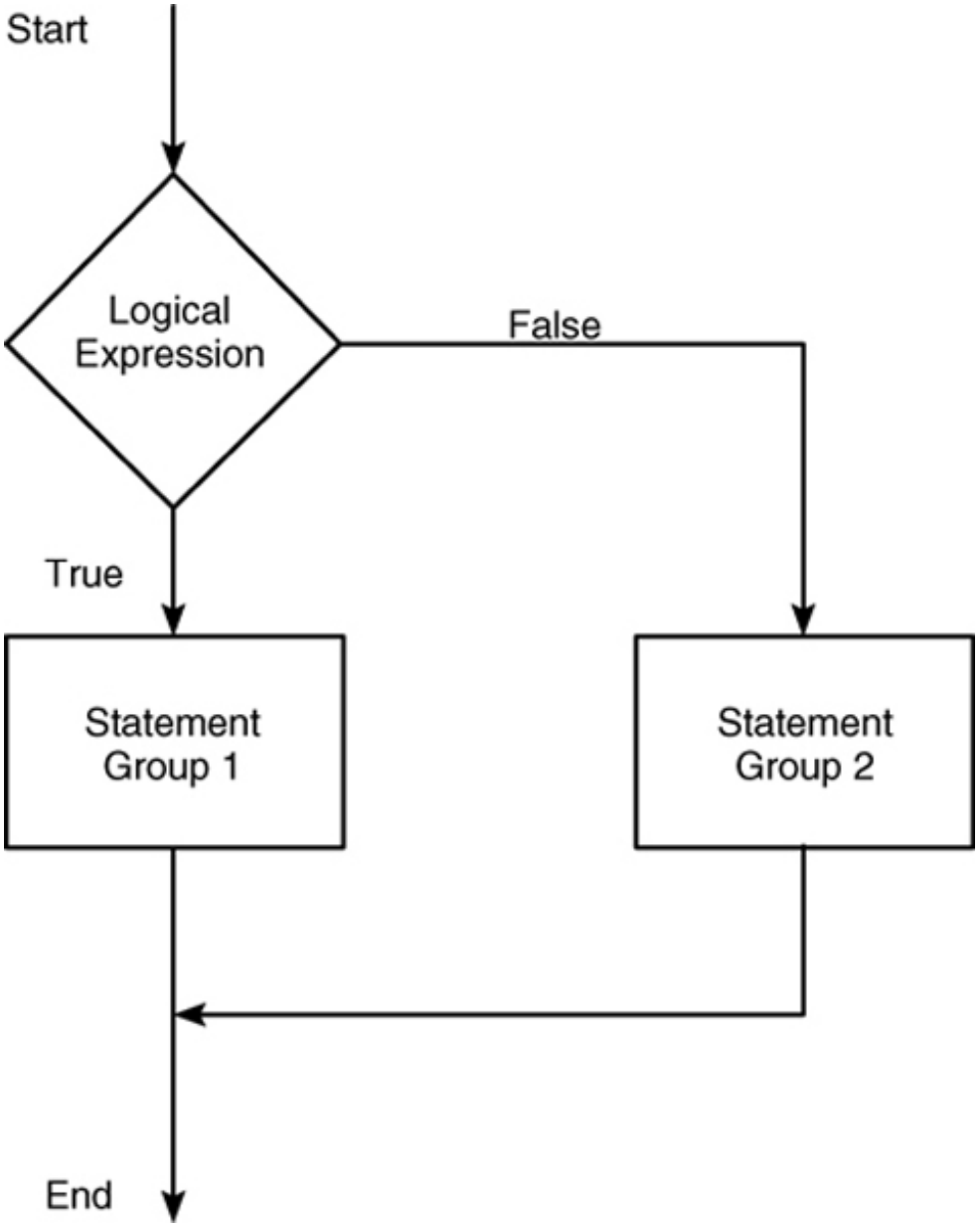
```
if cond1
    commands1
elseif cond2
    commands2
else
    commands3
end
```

- Use relational operators: ==, ~=, >, <, >=, <=, & and | (for elements),|| and && (scalars) , ~, xor, all, any

- No need to use parentheses in the conditions

IF Flowchart

- If the logical expression is a vector or matrix, the test returns a value of true only if **all** the elements of the logical expression are true!



Logical Expressions: example

- `>> x = [4,-9,25];`

```
if x < 0  ; true if all elements are negative
   disp('All of the elements of x are negative.')
else
   y = sqrt(x)
end
```

- Matlab compares x with 0, and calculates a vector of binary values: [0 1 0]. The result is false because 2 elements are not negative. So Matlab will skip the first command and program perform the code after else.
- When this program is run it gives the result

```
y =   2    0 + 3.000i    5
```

- If x = [-4, -9,-25] (for example), then the binary result would be [1 1 1] and it would display All of the elements of x are negative