

---

# CS1371

## Introduction to Computing for Engineers

### *Control Statements*

1

9/4/2003

---

## Control Flow Statements

### **Learning Objectives**

Learn about how to control the sequence of expressions that are evaluated in a program.

### **Topics**

- **IF** Statement and Logical Operators
- **SWITCH-CASE**
- **TRY-CATCH**
- **DISP ( )** vs **FPRINTF ( )**
- Statement display formats
- Special values
- Some useful Matlab built-in functions
- Summary

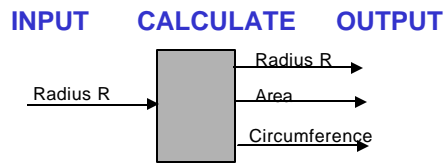
2

## Problem 1: Introducing the IF Structure

- **Step 1: Describe Problem:** Write a MatLab program to:

- enter the radius as an input variable
- calculate the area and the circumference of a circle
- output radius, area and circumference **IF** the area is greater than 20 square units.

- **Step 2: Describe input and output**



- **Step 3: Define test cases**

- $R=0 \rightarrow \text{Area}=0$  and  $\text{Circumference}=0$
- $R=1 \rightarrow \text{Area}=\pi$  and  $\text{Circumference}=2\pi$

3

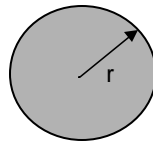
## Problem 1 ... contd.

- **Step 4: Develop the Solution**

- Describe the algorithm:

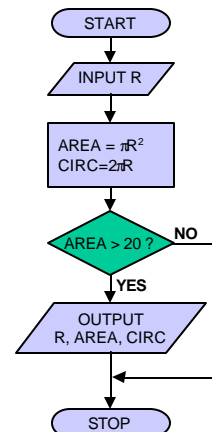
$$\text{Area} = \pi r^2$$

$$\text{Circumference} = 2\pi r$$



- Develop the process:

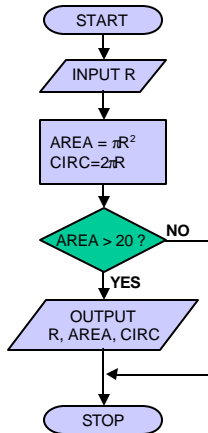
- Calculate the area
- Calculate the circumference
- If the area is big enough,
  - Display radius, area and circumference
  - otherwise, display nothing



4

## Problem 1 ... contd.

- **Step 4: Develop the Solution (cont'd)**  
– *Write Matlab code*



```
% calculate the area and circumference of a circle
% print it out if the area is greater than 20
radius = input('please enter a radius: ');
area = pi * radius^2;
circumference = 2 * pi * radius;
if area > 20
    fprintf('\n Radius = %f units', radius);
    fprintf('\n Area = %f units squared', area);
    fprintf('\n Circumference = %f units', circumference);
end
```

5

## Problem 1 ... contd.

- **Step 5: Test the Results**

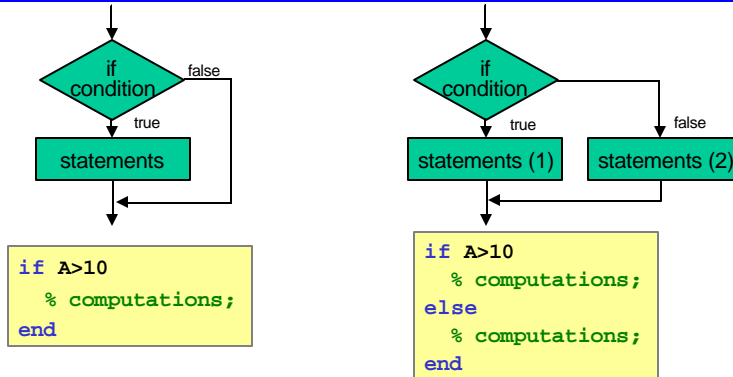
```
>> circle
please enter a radius: 0
>> circle
please enter a radius: 3

Radius = 3.000000 units
Area = 56.548668 units squared
Circumference = 18.849556 units
>> circle
please enter a radius: -5

Radius = -5.000000 units
Area = 157.079633 units squared
Circumference = -31.415927 units
>>
```

6

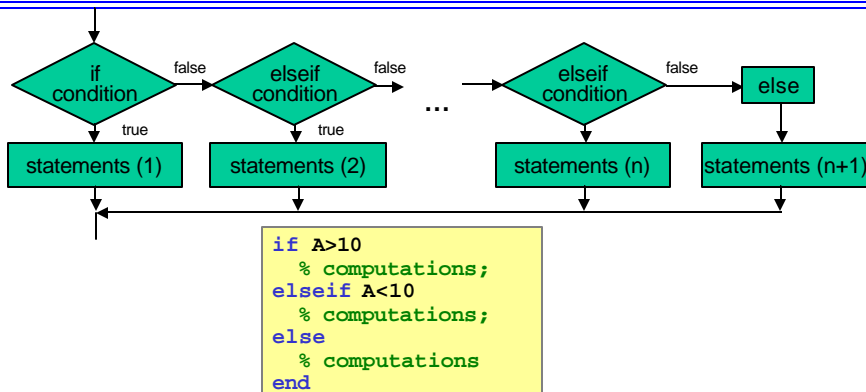
## IF statements



- Can include multiple statements
- Statements can also include other if statements (can nest if statements inside if statements)
- Be careful not to overlap (crossover) if statements!

7

## IF-ELSEIF statement



- Can have several **elseif** conditions...
- **else** is optional and executes if all other tests fail

8

## IF Statement and Logical Operators

### Relational Operators

<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equality
~=	Not equal

Start with a value for K:

```
K=5
```

Interpret the following in words

```
K>10
```

```
K*0 ~= 6
```

What if K is an array?

```
K=ones(5,5)
```

All elements in K are tested

```
if K>10
```

will fail, but

```
K(2,3)=20;
```

```
if K>10
```

will also fail because ALL elements must be >10.

9

## IF Statement and Logical Operators ...cont'd

### Logical Operators

OP	Symbol
not	~
and	&
or	
xor	xor(A,B)

**Note:** 0 is false

Any other numeric value is true

A	B	~A	A B	A&B	A xor B
0	0	1	0	0	0
0	1	1	1	0	1
1	0	0	1	0	1
1	1	0	1	1	0

10

## Switch-Case Statement

- If tests of a single expression must be made for many different conditions, a **SWITCH-CASE** statement may be a good choice
- Multiple conditions are contained in a **cell array**

```
switch expression
case condition_1
    %Do Stuff #1
case {condition_2a, condition_2b,...}
    %Do Stuff #2
...
otherwise
    %Do Other Stuff
end
```

11

## SWITCH-CASE Statement Example

```
% Demo of SWITCH-CASE control statement

x = input('Enter measurement in cm: ');
units = input('What output units? ', 's');

switch units
case {'inch','in'}
    y=x./2.54;
    fprintf('Result is: %f inches.',y)
case {'feet','ft'}
    y=x./2.54./12;
    fprintf('Result is: %f feet.',y)
case {'meter','m'}
    y=x./100;
    fprintf('Result is: %f meters.',y)
case {'centimeter','cm'}
    y=x;
    fprintf('Result is: %f centimeters.',y)
otherwise
    disp(['Unknown units: ' units])
    y=NaN;
end
```

input( ) used to read  
in a string...

```
>> basicSWITCH
Enter measurement in cm: 123
What output units? ft
Result is: 3749.040000 feet.
>> basicSWITCH
Enter measurement in cm: 123
What output units? ft
Result is: 4.035433 feet.
>> basicSWITCH
Enter measurement in cm: 123
What output units? m
Result is: 1.230000 meters.
>>
```

NEW: use [ ] in disp  
to concatenate strings

12

## TRY-CATCH Block

- When Matlab encounters an error, it displays an error message and control returns to the Command prompt
- What if you want to:
  - *trap the error in your program so that you can...*
  - *take corrective action without stopping execution*

```
try
    (commands)
catch
    (commands if error occurs in commands)
end
```

13

## TRY-CATCH Block Example

```
x=ones(4,2); % create two test matrices
y=4*eye(3);
try
    z=x*y % matrix multiplication
catch
    z=NaN;
    disp('X and Y are not conformable matrices')
end
z % display or use result in rest of program
```

```
>> basicTRY
X and Y are not conformable matrices

z =

    NaN

>>
```

- Program will not terminate if matrices cannot be multiplied
  - *instead will set product to NaN*
  - *display a message*
  - *continue execution*
- Can have multiple TRY-CATCH blocks in a program
- Similar code is used in other languages for this purpose

14

## Display **FORMAT**

Enter the following Matrix into Matlab...

```
M = [55.3 -22 12; 10 23.4324 30.42]
```

• Explore each format:

- | • <b>COMMAND</b>              | <b>FUNCTION</b>    |
|-------------------------------|--------------------|
| • <code>format short</code>   | <b>default</b>     |
| • <code>format long</code>    | <b>14 decimals</b> |
| • <code>format short e</code> | <b>4 decimals</b>  |
| • <code>format long e</code>  | <b>15 decimals</b> |
| • <code>format bank</code>    | <b>2 decimals</b>  |
| • <code>format +</code>       | <b>+,-,blank</b>   |

15

## **DISP()** and **FPRINTF()**

- `disp(X)` – prints elements of an array **X**
- `disp('hello world')` – prints the string
- `fprintf(fid, format, A)` – does the following:
  - 1. Write **A** to file **fid** using **format** (omitting **fid** prints to screen)
  - 2. **format** is a string containing output string and format instructions for each variable in **A**
  - 3. Variables of all printable data types:  
Conversion specifications involve the character %, optional flags, optional width and precision fields, optional subtype specifier, and conversion characters: **d, i, o, u, x, X, f, e, E, g, G, c, and s.**
  - 4. The special formats **\n, \r, \t, \b, \f** can be used to produce *linefeed, carriage return, tab, backspace, and formfeed characters respectively.*
- See `help fprintf` and `help disp` for more information about these useful functions...

16

## REVIEW: `INPUT( )` for Getting User Input

- How do you prompt user for input?

```
Myvariable = input('Some String');
```

- What if you actually want the string the user enters?

```
Myvariable = input('Another String', 's');
```

- Explore the use of these functions; check out `help input` for more details...

17

## REVIEW: Special Values

These objects have special meanings in Matlab:

`pi` the value 3.1416 (*How would show more values?*)

`i, j` `sqrt(-1)`

`inf` infinity (*How can you prove this represents infinity?*)

`NaN` “Not a Number” (*When do you get this message?*)

`clock` matrix with date and time

`date` Current date in string form

`eps` “epsilon” is the smallest amount by which two floating point values can differ on the current computer

`ans` value just computed if not assigned

18

## Built-In Functions

- Matlab includes all of the trig functions and their inverses (you may want to explore the `tan( )` and `atan( )` and `atan2( )` functions which can be tricky.

- If X is a vector, then what do these functions do? What if X is an array?

```
max(X)
min(X)
mean(X)
median(X)
sum(X)
prod(X)
cumsum(X)
cumprod(X)
std(X)
```

19

## Matlab Built-In Functions ... *cont'd*

**On Your Own:** Let's experiment with some functions and see if you can explain how the following functions operate!

- `round(-2.6)`
- `fix(-2.6)`
- `floor(-2.6)`
- `ceil(-2.6)`
- `sign(-2.6)`
- `rem(15,2)` %Doesn't "mod" do the same thing?
- `floor(ceil(10.8))`
- `log10(100)` %What's the base of "log"?
- `abs(-5:5)`
- `round([0:0.3:2,1:0.75:4])`

20

## Built-In Functions (Cont.)

• These functions generate random numbers. Can you figure out the differences?

- `randn(n)`
- `rand(n)`
- `rand(m,n)`
- `rand('seed',n)`
- `rand('seed')`

21

## Summary

### • Topics Covered

- `if` Statement and Logical Operators
- `switch-case`
- `disp()` vs `fprintf()`
- `input()`
- Statement Display Format
- Special Values
- Some useful Matlab Built-in Functions
- Summary

### Action Items

- Review the lecture
- Keep exploring MATLAB!
- Come prepared to ask questions about MATLAB

22

---

**Questions?**