“if ... end” structure

The “IF” statement

The if statement chooses whether another statement, or group of statements, is executed or not. The general form of the if statement is:

```
if condition

action

end
```

For example, the following if statement checks to see whether the value of a variable is negative. If it is, the value is changed to a zero; otherwise, nothing is changed.

```
if num < 0

num = 0

end
```

sqrtifexmp.m

```
% Prompt the user for a number and print its sqrt
num = input ( ' Please enter a number: ' ) ;
```

```
The number entered is : -4.5
```

```
fprintf("The number entered is : %.1f",num)
% If the user entered a negative number, change it
if num < 0
num = 0 ;
end
fprintf ( 'The sqrt of : %.1f is : %.1f\n' , num, sqrt (num))
```

```
The sqrt of : 0.0 is : 0.0
```

sqrtifexampii .m

```
% Prompt the user for a number and print its sqrt
num = input ( ' Please enter a number: ' ) ;
fprintf("The number entered is : %.1f",num)
```

```
The number entered is : -25.0
```

```
% If the user entered a negative number, tell the user and change it
if num < 0
    disp('OK, we''ll use the absolute value' )
    num = abs (num);
end
```

```
OK, we'll use the absolute value
```

```
fprintf ( 'The sqrt of %.1f is %.1f\n' , num, sqrt(num) )
```

```
The sqrt of 25.0 is 5.0
```

createvec.m

```
function outvec = createvec (mymin, mymax)
% createvec creates a vector that iterates from a
% specified minimum to a maximum
% Format of call: createvec (minimum, maximum)
% Returns a vector If the "minimum" isn't smaller than the "maximum",
% exchange the values using a temporary variable
if mymin > mymax
    temp = mymin;
    mymin = mymax;
    mymax = temp ;
end
% Use the colon operator to create the vector
outvec = mymin : mymax ;
end
```

Output

```
createvec(4,10)

ans =

4     5     6     7     8     9    10
```

The “if-else” Statement

The if statement chooses whether or not an action is executed. Choosing

between two actions, or choosing from among several actions, is accomplished

using if-else, nested if-else, and switch statements.

The if-else statement is used to choose between two statements, or sets of statements. The general form is:

if condition

action1

else

action2

end

For example, to determine and print whether or not a random number in the

range from 0 to 1 is less than 0.5, an if-else statement could be used:

```
if rand < 0.5
    disp ( 'It was less than . 5!' )
else
    disp ( ' It was not less than . 5! ')
end
```

```
It was not less than . 5!
```

One application of an if-else statement is to check for errors in the inputs to a script (this is called error-checking). For example, an earlier script prompted the user for a radius, and then used that to calculate the area of a circle. However, it did not check to make sure that the radius was valid (e.g., a positive number). Here is a modified script that checks the radius:

checkradius.m

```
% This script calculates the area of a circle
% It error-checks the user's radius
radius = input ( ' Please enter the radius: ') ;
if radius <= 0
    fprintf ( 'Sorry; %.2f is not a valid radius\n' , radius)
else
    area = pi*radius*radius ;
    fprintf ( ' For a circle with a radius of %.2f, ' , radius)
    fprintf ( ' the area is : %.2f\n' , area)
end
```

```
For a circle with a radius of 12.00,
the area is : 452.39
```

Nested “if-else” Statement

y =1 if x < -1

y = x^2 if - 1 <= x <=2

y = 4 if x > 2

The value of y is based on the value of x, which could be in one of three possible

ranges. Choosing which range could be accomplished with three separate if

statements, is as follows:

```
x = 2;
if x < -1
    y= 1;
end
if x >= -1 && x <=2

    y = x^2;
end
if x > 2
```

```
    y = 4;
end
fprintf("y = %d",y)
```

```
y = 4
```

“elseif” Statement

For example, the previous example could be written using the elseif clause

rather than nesting if-else statements:

```
if x < -1

y = 1;

elseif x <= 2

y = x^2;

else

y = 4;

end
```

calcy.m

```
function y = calcy (x)
%  calcy calculates y as a function of x
%  Format of call: calcy (x)
if x <-1
    y = 1;
elseif x <= 2
    y = x^2;
else
    y = 4;
end
end
```

Output

```
x = 1.1

x =

1.1000

y = calcy(x)

y =

1.2100
```

findargtype .m

```
function outtype = findargtype(inputarg)
% findargtype determines whether the input
% argument is a scalar, vector, or matrix
% Format of call: findargtype (inputArgument)
% Returns a string

[r c] = size (inputarg) ;
if r==1 && c ==1
    outtype = 'scalar' ;
elseif r==1 || c==1
    outtype = 'vector' ;
else
    outtype = 'matrix';
end
end
```

Output

```
findargtype([2,4,5])


ans =


'vector'


findargtype(2)


ans =


'scalar'
```

letgrade.m

```
% function grade = letgrade (quiz)
% % letgrade returns the letter grade corresponding
% % to the integer quiz grade argument
% % Format of call: letgrade (integerQuiz)
% % Returns a character
% % First, error-check
% if quiz < 0 || quiz > 10
%     grade = 'x' ;
% % If here, it is valid so figure out the
```

```
% % corresponding letter grade
% elseif quiz == 9 || quiz == 10
%     grade = 'A' ;
% elseif quiz == 8
%     grade = 'B' ;
% elseif quiz == 7
%     grade = 'c' ;
% elseif quiz == 6
%     grade = 'D' ;
% else
%     grade = 'F' ;
% end
% end
```

Output

```
quiz = 8


quiz =


8


lettergrade = letgrade(quiz)


lettergrade =


'B'


quiz = 4


quiz =


4


lettergrade = letgrade(quiz)


lettergrade =
```

```
'F'
```

Practice 4.1

Write an if statement that would print "Hey, you get overtime!" if the value of a variable hours is greater than 40. Test the if statement for values of hours less than, equal to, and greater than 40. Will it be easier to do this in the Command Window or in a script?

```
% Enter variable hours
variableHours = input('Enter Variable Hours : ');
% Variable Hours greater than 40
if variableHours > 40
    fprintf("The hours worked are : %d",variableHours)
    fprintf('Hey! you get overtime! \n');
end
```

```
The hours worked are : 50
Hey! you get overtime!
```

Practice 4.3

Modify the function findargtype to return ‘scalar’, ‘row vector’, ‘column vector’, or ‘matrix’ depending on the input argument

```
% function type = findargtype2(X)
%
% findargtype2 finds if the input is a scalar, row vector,
%
% column vector, or matrix.
%
% Format of call: findargtype2(X)
%
% Returns a string
%
% [r c] = size(X);
%
% if r == 1 && c == 1
%
% type = 'scalar';
%
% elseif r == 1
%
% type = 'row vector';
%
% elseif c == 1
%
% type = 'column vector';
%
```

```
% else
%
% type = 'matrix';
%
% end
%
% end
```

Output

```
findargtype2([2;4;5])


ans =


'column vector'


findargtype2([2,4,5])


ans =


'row vector'
```

Practice 4.4

Modify the original function findargtype to use three separate if statements instead of a nested if-else statement.

```
% function outtype = findargtype_prac(inputarg)
% findargtype determines whether the input argument is
%a scalar, vector or matrix and returns a string
[r, c] = size(inputarg);
if r == 1 && c == 1
    outtype = 'scalar';
end
if r == 1 && c > 1
    outtype = 'vector';
end
if r > 1 && c > 1
    outtype = 'matrix';
end
end
```

Output

```
findargtype_prac(2)


ans =


'scalar'
```

Practice 4.2

Write a script printsindegorrad that will:

1. Prompt the user for an angle. :

2. Prompt the user for (r)adians or (d)egrees, with radians as the default. :

3. If the user enters ‘d’, the sind function will be used to get the sine of the angle in degrees; otherwise, the sin function will be used. Which sine function to use will be based solely on whether the user enters a ‘d’ or not. A ‘d’ means degrees, so sind is used; otherwise, for any other character the default of radians is assumed so sin is used. :

4. Print the result.

Here are examples of running the script:

>> printsindegorrad

Enter the angle: 45

(r)adians (the default) or (d)egrees: d

The sin is 0.71

>> printsindegorrad

Enter the angle: pi

(r)adians (the default) or (d)egrees: r

The sin is 0.00

```
% This script shows how to use an if-else statement

% Ask for the angle

ang = input('Enter the angle: ');

% Ask for the units

u = input('Enter d for degrees or r for radians: ','s');

% Test to select which sin function to use, sind or sin
```

```
if u == 'd'

res = sind(ang);

else

res = sin(ang);

end

% Display the answer

fprintf('sin(%.2f) = %.2f\n',ang,res)
```

Output

```
printsindegorrad

Enter the angle: 45

Enter d for degrees or r for radians: r

sin(45.00) = 0.85

printsindegorrad

Enter the angle: pi

Enter d for degrees or r for radians: r

sin(3.14) = 0.00
```