AYDIN ADNAN MENDERES UNIVERSITY FACULTY OF ENGINEERING

Department of Electrical and Electronics Engineering

EE213 – Transform Techniques With Computer Applications

2022-2023, Fall

Presentation 5

Dr. Adem Ükte

Selection Structures

If Structures

The **if** selection structure can be used in three forms:

I

```
if expression
statements
end
```

II

```
if expression
statements
else
statements
end
```

III

```
if expression
statements
elseif expression
statements
        .
        .
        .
else
statements
end
```

In the first form, an ***expression*** is evaluated, and a group of ***statements*** are executed when the ***expression*** is true. An ***expression*** is true when its result is nonempty and contains only nonzero elements (logical or real numeric). Otherwise, the ***expression*** is false. If the ***expression*** is false, the program jumps immediately to the next row after **end**.

In fact, the ***expression*** does not have to be a ***comparison***. However, ***comparison*** is often used as ***expression*** in **if** structure.

The **elseif** and **else** blocks in the second and third forms are optional. The ***statements*** are executed only if previous expressions in the **if** ... **end** block are false.

An **if** block can include multiple **elseif** blocks.

Selection Structures

If Structures

A simple example of an **if** statement:

```
G=input('Please enter a value for G');
if G<50
disp('G is a small value equal to:')
disp(G);
end
```

If **G** is less than 50, then the statements between **if** and **end** lines are executed.

For example, if **G** has a value of 25, then the below output will be shown in command window:

```
G is a small value equal to:
    25
```

Selection Structures

If Structures

Here is an example of an **if**/**else** statement:

```
x=input('Please enter an input parameter for log function');
if x >0
y = log(x);
fprintf('Natural logarithm of %1.2f is equal to %1.2f\n',x,y)
else
disp('The input to the log function must be positive')
end
```

If **x** is positive, then the statements between **if** and **else** lines are executed. For example, if **x=5**, then the below output will be shown in command window:

```
Natural logarithm of 5.00 is equal to 1.61
```

If **x** is negative, then the statements between **else** and **end** lines are executed and the below output will be shown in command window:

```
The input to the log function must be positive
```

Selection Structures

If Structures

Here is an example of an **if**/**elseif** /**else** statement:

```
x=input('Please enter a value for x ');
minVal = 2;
maxVal = 6;
if (x >= minVal) && (x <= maxVal)
    disp('Value within specified range.')
elseif (x > maxVal)
    disp('Value exceeds maximum value.')
else
    disp('Value is below minimum value.')
end
```

If **x** is between 2 and 6, then the statements between **if** and **elseif** lines are executed. For example, if **x=5**, then the below output will be shown in command window:

```
Value within specified range.
```

If **x** is greater than 6, then the statements between **elseif** and **else** lines are executed, and the below output will be shown in command window:

```
Value exceeds maximum value.
```

If **x** is less than 2, which is the all other possibilities, then the statements between **else** and **end** lines are executed, and the below output will be shown in command window:

```
Value is below minimum value.
```

Selection Structures

Switch/Case Structures

In this structure, one of several groups of statements is executed.

```
switch switch_expression
   case case_expression
      statements
   case case_expression
      statements
    ...
   otherwise
      statements
end
```

An expression is evaluated and one of several groups of statements is choosen to execute. Each choice is a case.

The switch block tests each case until one of the case expressions is true.

A case is true when **`case_expression == switch_expression`**.

When a case expression is true, MATLAB® executes the corresponding statements and exits the switch block.

The otherwise block is optional. MATLAB executes the statements only when no case is true.

Selection Structures

Switch/Case Structures

Here is an example:

```
n = input('Enter a number: ');
switch n
    case -1
        disp('negative one')
    case 0
        disp('zero')
    case 1
        disp('positive one')
    otherwise
        disp('other value')
end
```

Based on the value of variable n entered by the user, an appropriate **`case`** or **`otherwise`** block will be executed.

For example, if **`n`** is entered as **`1`**, **`positive one`** string will be displayed on command window.

If **`n`** is entered as **`2`**, which is not equal to **`-1`**, **`0`** or **`1`**, **`otherwise`** block will be executed, and **`other value`** string will be displayed on command window.

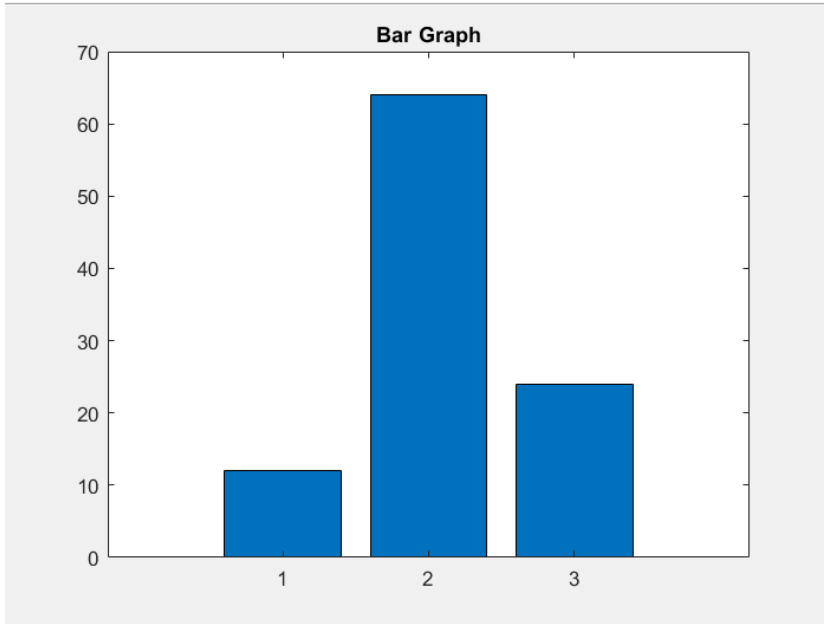
Selection Structures

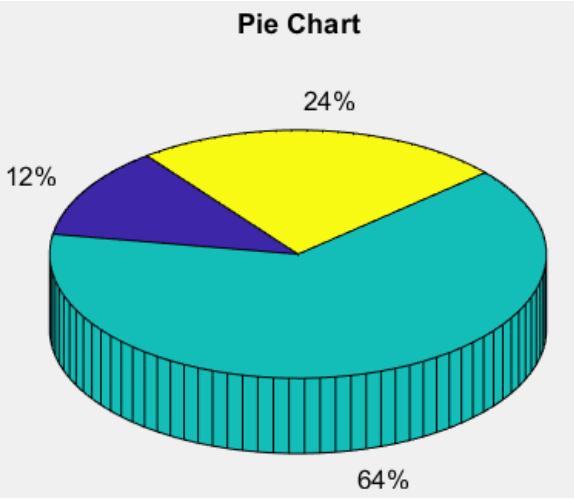
Switch/Case Structures

There can be multiple choices in **case** expressions:

```
x = [12 64 24];
plottype = input('Please enter the plot type: ','s');
switch plottype
    case 'bar'
        bar(x)
        title('Bar Graph')
    case {'pie','pie3'}
        pie3(x)
        title('Pie Chart')
    otherwise
        warning('Unexpected plot type. No plot created.')
end
```

In this example, if the value of **plottype** variable is entered as either **pie** or **pie3** string by user, the statements in the second **case** block will be executed and 3D pie graph of **x** vector will be drawn.






```
Warning: Unexpected plot type. No plot created.
```

Repetition Structures

Repetition structures are often called loops. All loops consist of five basic parts:

- ✓ A parameter to be used in determining whether or not to end the loop
- ✓ Initialization of this parameter
- ✓ A way to change the parameter each time through the loop. (If you don't change it, the loop will never stop executing.)
- ✓ A comparison (a criterion used to decide when to end the loop)
- ✓ Calculations to do inside the loop

For Loops

```
for index = values
   statements
end
```

The first line identifies the loop and defines an index, which is a number that changes on each pass through the loop and is used to determine when to end the repetitions.

After the identification line, it comes statements (the group of commands) to be executed.

Finally, the end of the loop is identified by the command **`end`**.

The loop is executed once for each value of the index variable identified in the first line.

For Loops

Here's a simple example:

```
for k = [1,3,7]
k
end
```

- During the first pass through the loop, **k** is assigned to a value of **1**, the first value in the **k** matrix.
- Then the command(s) between **for** and **end** lines are executed.
- During the next pass, the value of **k** is modified to **3**, the second value in the **k** matrix.
- Then again, the command(s) between **for** and **end** lines are executed.
- During the last pass, the value of **k** is modified to **7**, the third value in the **k** matrix.
- Then again, the command(s) between **for** and **end** lines are executed for the last time.
- Each time through the loop, **k** is modified and assigned to subsequent values from the index matrix.

This example code returns:

```
k =
     1
k =
     3
k =
     7
```

For Loops

In most cases, index vector is defined with start point, end point, and increment/decrement value if necessary:

```
for i = 1:3
a = 5^i
end
```

The above code returns:

```
a =
     5
a =
    25
a =
   125
```

For Loops

Here's another example:

```
for i = 1:5
a(i) = i^2
end
```

It returns:

```
a =
     1
a =
     1     4
a =
     1     4     9
a =
     1     4     9    16
a =
     1     4     9    16    25
```

While Loops

```
while expression
    statements
end
```

In while loop, an **`expression`** is evaluated, and the execution of a group of **`statements`** in a loop is repeated while the **`expression`** is true.

An expression is true when its result is nonempty and contains only nonzero elements (logical or real numeric).

Otherwise, the expression is false.

Here's an example:

```
k = 0;
while k<3
k = k+1
end
```

```
k =
     1
k =
     2
k =
     3
```

In this case, we initialized a counter, **`k`**, before the loop.

Then the loop repeated as long as **`k`** was less than 3.

We incremented **`k`** by 1 every time through the loop, so that the loop repeated three times.

While Loops

Here's another example:

```
n = 10;
f = n;
while n > 1
    n = n-1;
    f = f*n;
end
disp(['10! = ' num2str(f)])
```

It returns:

```
10! = 3628800
```

NesteD Loops

It is often useful to nest loops inside other loops.

Nested loops are oftenly used for looking inside the matrices.

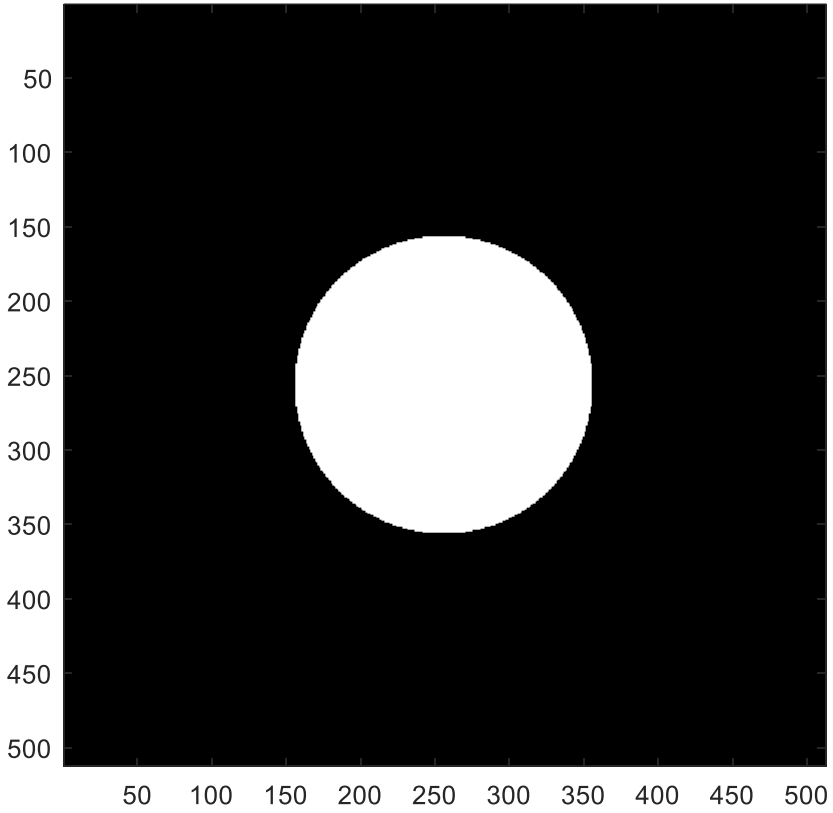
For example, we can use the following code to change the values of elements of a matrix whose value is greater than 5 to 5.

```
x=[1,7,3;6,8,-2;5.1,5.01,5.001];
disp('Old x matrix:')
disp(x)
for i=1:size(x,1)
    for j=1:size(x,2)
        if x(i,j)>5
            x(i,j)=5;
        end
    end
end
disp('New x matrix:')
disp(x)
```

```
Old x matrix:
    1.0000    7.0000    3.0000
    6.0000    8.0000   -2.0000
    5.1000    5.0100    5.0010
New x matrix:
     1     5     3
     5     5    -2
     5     5     5
```

Practice 8:

In this practice, an image will be obtained consisting of a white circle with a center of (256,256) pixels and a radius of 100 pixels located inside a black square of 512x512 pixels.



- The equation in the $x-y$ plane of a circle of radius r with its center at (a, b) point is given below:

$$(x-a)^2+(y-b)^2=r^2$$

```
clc;clear;close all;
a=256; b=256;%Center coordinates of the circle
M=512;%Number of rows
N=512;%Number of columns
I=zeros(M,N);%Preallocation
for x=1:M
    for y=1:N
        radius=sqrt((x-a)^2+(y-b)^2);
        if radius<100
            I(x,y)=1;
        else
            I(x,y)=0;
        end
    end
end
imshow(I)
```

Practice 9:

Write a program that checks whether a number entered by the user is prime or not.

As you know, a prime number is a number that is only divisible by 1 and itself. In other words, **a number is not prime if it has divisor(s) other than 1 and itself**.

You will follow the below steps to write the program:

- ✓ Ask the user to enter a number.
- ✓ If the user enters a number less than 3 or a decimal (non-integer) number, the program should first print a **warning** text on the screen and then ask the user to enter a correct number again. The process of printing this warning and asking the user to re-enter the number must **repeat as long as the user enters an integer number that is grater than or equal to 3**.
- ✓ After the user has entered the correct number, you will use a variable as the prime flag when checking if the number is prime. You will initially assume that the number is prime. So, assign this variable a constant value representing its prime case (for example, 1).
- ✓ Now for all numbers from 2 to one less than the number entered by the user, you should check one by one whether the number entered by the user is **exactly divisible** by this number. If the number entered by the user is exactly divisible by any of the numbers in this range, the original assumption of being prime is no longer valid and you will need to assign the variable specifying the prime flag a different constant than the previous one representing the non-prime case (for example, 0).
- ✓ Finally, you should print on the screen that the number entered is prime or non-prime, depending on whether the value of the variable specifying the prime flag is equal to the constant number representing the prime case (1) or not (0).

Practice 9:

```
%EE213 2022-2023 Fall
%Practice 9: program that checks whether a number entered by the user
%is prime or not
clc,clear,close all
a=input('Please enter an integer number that is grater than or equal to 3: ');
while a<3 || fix(a)~=a
    warning('You should enter an integer number that is grater than or equal to 3!')
    a=input('Please enter a correct number again: ');
end
prime_flag=1;%Initially we assume prime (Prime flag)
for i=2:a-1
    if rem(a,i)==0
        prime_flag=0;
    end
end
if prime_flag==1
    fprintf('The number %d is prime.\n',a)
else
    fprintf('The number %d is not prime.\n',a)
end
```

END OF PRESENTATION 5

References:

✓MathWorks® Help Center, https://www.mathworks.com/help/.