SELECTION STRUCTURES*

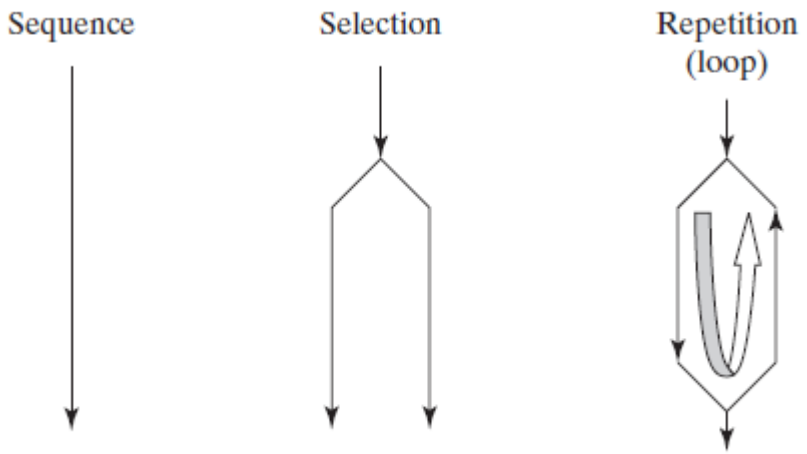
*** Some of the contents are adopted from**
MATLAB® for Engineers, Holly Moore, 3rd Ed., Pearson Inc., 2012.

Control Structures

- One way to think of a computer program (not just MATLAB) is to consider how the statements that compose it are organized.
- Usually, sections of computer code can be categorized as ***sequence structure*** , ***selection structure*** , and ***repetition structure***

3 control structures:

1. sequence structure,
2. selection structure,
3. repetition structure.



Control Structures ctd.

1. A sequence is a list of commands that are executed one after another. (So far, we have written code that contains sequences)

2. A selection structure allows the programmer to execute one command (or set of commands) if some criterion is true and a second command (or set of commands) if the criterion is false. A selection statement provides the means of choosing between these paths, based on a ***logical condition*** . The conditions that are evaluated often contain both ***relational*** and ***logical*** operators or functions.
3. A repetition structure, or loop, causes a group of statements to be executed multiple times. The number of times a loop is executed depends on either a counter or the evaluation of a logical condition.

Relational Operators, Revisited

If we define two scalars

```
x = 5;
y = 1;
```

and use a relational operator such as <, the result of the comparison

```
x<y
```

is either true or false. In this case, **x** is not less than **y** , so MATLAB responds

```
ans =
0
```

indicating that the comparison is false. MATLAB uses this answer in selection

statements and in repetition structures to make decisions.

Relational Operators, ctd.

We can also see how MATLAB handles comparisons between matrices.

```
x = 1:5;
y = x -4;
x<y
```

returns

```
ans =
0 0 0 0 0
```

MATLAB compares corresponding elements and creates an answer matrix of zeros and ones. In the preceding example, **x** was greater than **y** for every comparison of elements, so every comparison was false and the answer was a string of zeros. If, instead, we have

```
x = [ 1, 2, 3, 4, 5];
y = [-2, 0, 2, 4, 6];
x<y
```

then

```
ans =
0 0 0 0 1
```

Relational Operators, ctd.

Operator	Description
==	Equal to
~=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Operation	Result
5==6	0
‘A’<‘Z’	1
5>6	0
5>=6	0
5<6	1
6<=6	1

The operators <, >, <=, and >= use only the real part of their operands for the comparison. The operators == and ~= test real and imaginary parts.

Logical Operators, Revisited

There are four logical operators in MATLAB whicah are “& (AND)”, “| (OR)”, “xor (XOR)” ve “~ (NOT)”.

Inputs (Logical)		**AND**	**OR**	**XOR**	**NOT**
a	**b**	**a&b**	**a|b**	**xor(a,b)**	**~a**
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

The functions of these operators are given below.

Operator	**Symbols**	**Example**	
and	&	and(1,0)=0	and(1,1)=1
or	|	or(0,0)=0	or(1,0)=1
not	~	not(5)=0	not(0)=1
xor	xor	xor(5,1)=0	xor(0,3)=1

Relational & Logical Operators

MATLAB also allows us to combine comparisons with the logical operators ***and*** , ***not*** , and ***or,*** *t*he code

```
x = [ 1, 2, 3, 4, 5];
y = [-2, 0, 2, 4, 6];
z = [ 8, 8, 8, 8, 8];
z>x & z>y
```

returns

```
ans =
1 1 1 1 1
```

because **z** is greater than both **x** and **y** for every element. The statement

```
x>y | x>z
```

is read as " **x** is greater than **y** or **x** is greater than **z** " and returns

```
ans =
1 1 1 0 0
```

Selection Structures

- This section describes the syntax used in if statements.

- Three selection forms

    1. Simple if statements (to execute one group of statements selectively)
    2. if-else statements (two execute two group of statements selectively)
    3. if-elseif-else statements (to execute multiple group of statements selectively)

if Statement

A simple if statement has the following form:

```
if (comparison)
   statements
end
```

If the *comparison* (a logical expression) is true, the statements between the if keyword and the end keyword are executed. If the comparison is false, the program jumps immediately to the statement following the end keyword.

if Statement, ctd.

A simple example of an if structure

```
if (G<50)
   disp('G is a small value equal to:')
   disp(G);
end
```

This if structure (from if to end) is easy to interpret if *G* is a scalar. If *G* is less than 50, then the statements between **`if`** and **`end`** are executed. For example, if G has a value of 25, then

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

is displayed on the screen. However, if **G** is not a scalar, then the **if** statement considers the comparison true **only if it is true for every element**! Thus, if G is defined as

```
G = 0:10:80;
```

the comparison is false, and the statements inside the if statement are not executed! In general, if statements work best when dealing with scalars.

if-else Statement

- The simple if allows us to execute a series of statements if a condition is true and to skip those steps if the condition is false.
- Here's a set of if-else statements that calculates the logarithm if the input is positive and sends an error message if the input to the function is 0 or negative:

```
if (x>0)
   y = log(x)
else
   disp('The input to the log function must be positive')
end
```

(a single group of statements are executed if the condition is true,

(another group of statements are executed if the condition is false)

When *x* is a scalar, this is easy to interpret. However, when *x* is a matrix, the comparison is true only if it is true for every element in the matrix. So, if

```
x = 0:0.5:2;
```

then the elements in the matrix are not all greater than 0. Therefore, MATLAB skips to the else portion of the statement and displays the error message.

if-elseif-else Statement

- When we nest several levels of if-else statements, it may be difficult to determine which logical expressions must be true (or false) in order to execute each set of statements.
- The elseif allows us to check multiple criteria while keeping the code easy to read. Consider the following lines of code that evaluate whether to issue a driver's license, based on the applicant's age:

```
if (age<16)                                  If statement related to the condition (age<16)
   disp('Sorry – You'll have to wait')
elseif (age<18)                              else if statement related to the condition (age<18)
   disp('You may have a youth license')
elseif (age<70)                              else if statement related to the condition (age<70)
   disp('You may have a standard license')
else                                         else statement
   disp('Drivers over 70 require a special            license')
end
```

if-elseif-else Statement, ctd.

- The conditions are evaluated in such an order;
- If a condition is true, the statement associated with it is executed, and this terminates the whole chain.
- As always, the code for each statement is either a single statement, or group of them in braces.
- The last else part handles the “none of the above” or default case where none of the other conditions is satisfied.

```
if (condition1)
  statement1;
elseif (condition2)
  statement2;
elseif (condition3)
  statement3;
else
  statement4;
end
```

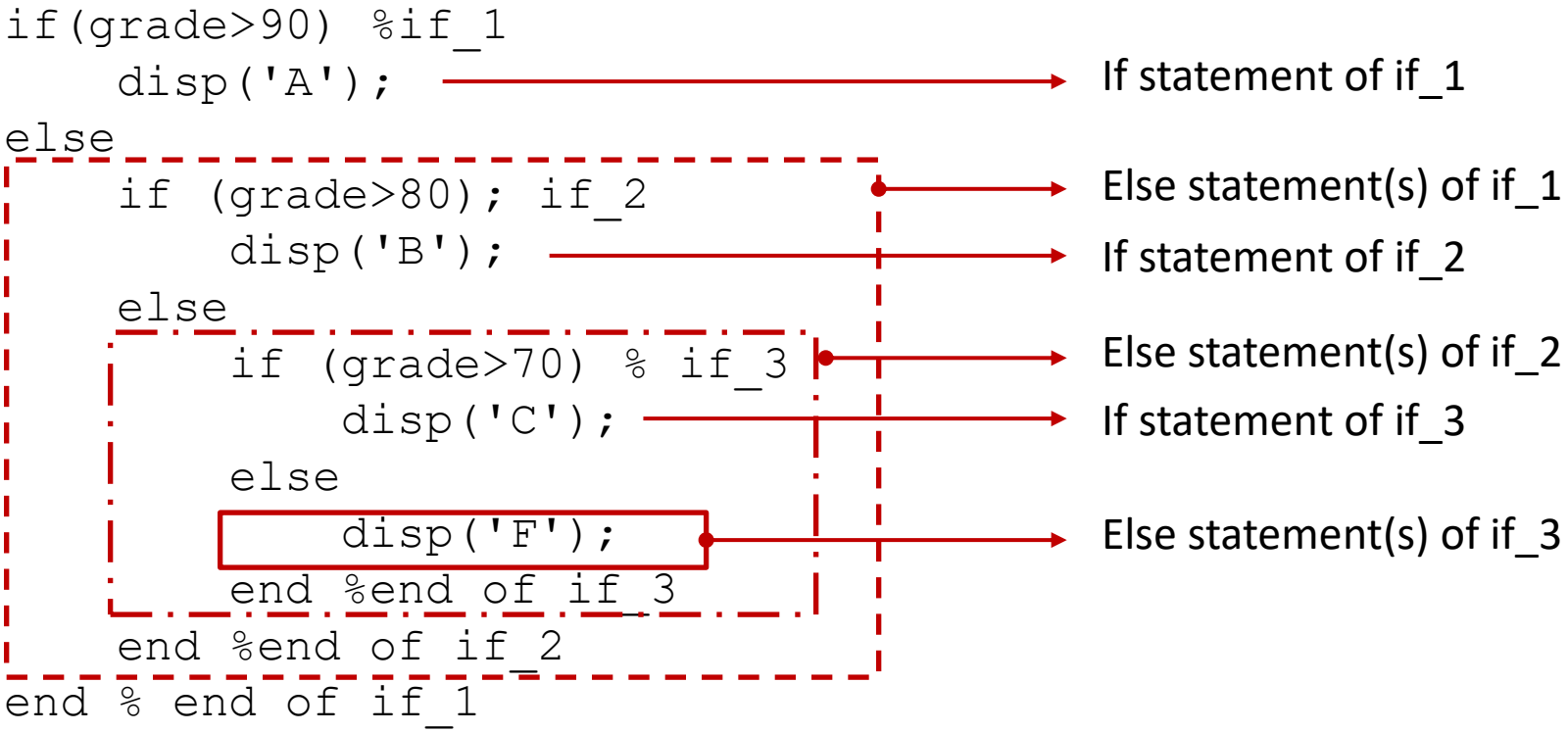
Selection Structures, ctd.

Two equivalent if structures,

```
if(grade>90)
   disp('A');
elseif (grade>80)
   disp('B');
elseif (grade>70)
   disp('C');
else
   disp(‘F’);
end
```

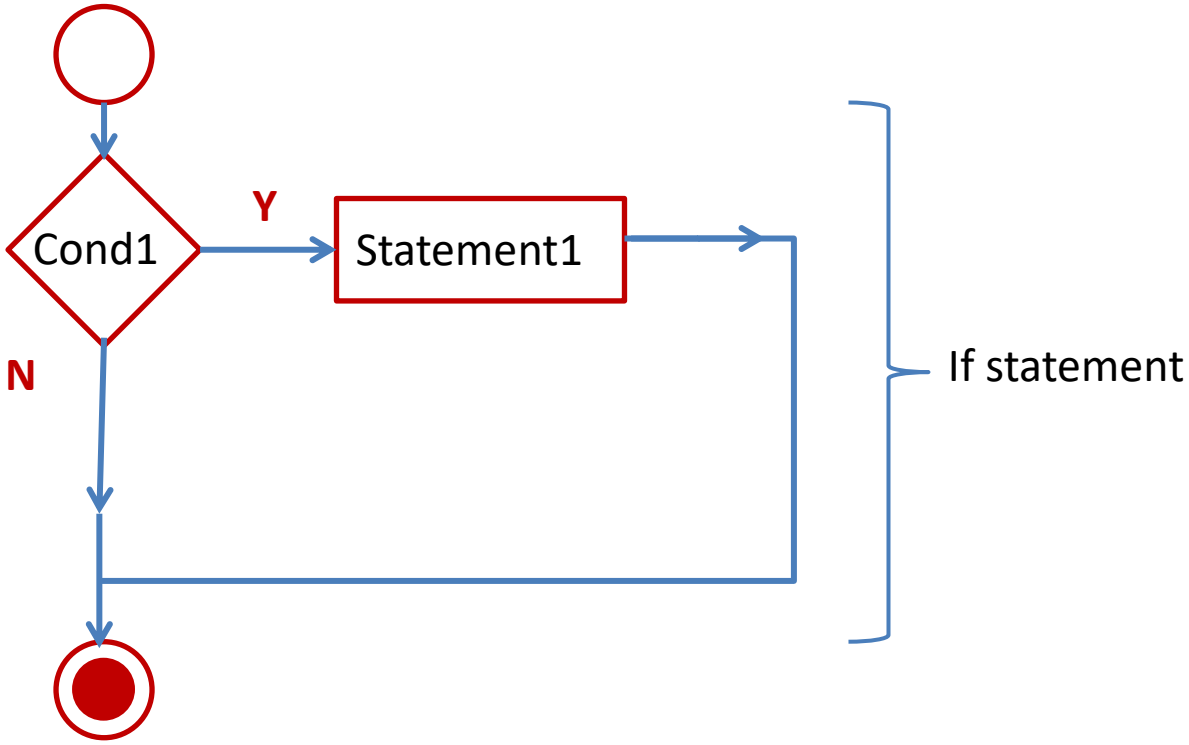
```
if(grade>90) %if_1
   disp('A');
else
   if (grade>80);
      disp('B');
   else
      if (grade>70) % if_3
         disp('C');
      else
         disp('F');
      end %end of if_3
   end %end of if_2
end % end of if_1
```

Selection Structures, ctd.



Selection Structures, ctd.

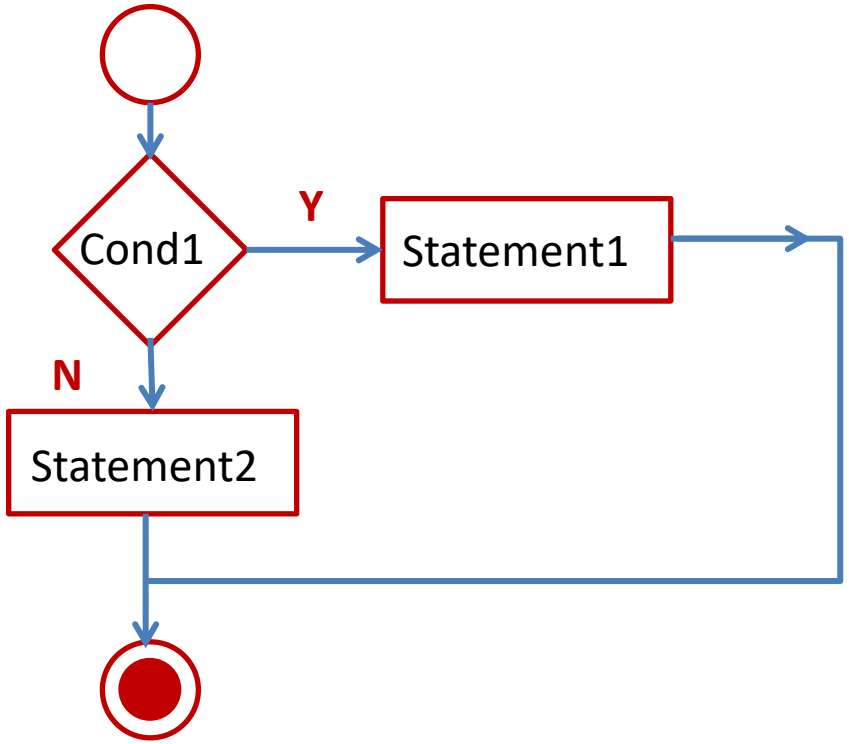
Flow-chart



- A diamond indicates a decision point
- Calculations are placed in rectangles
- Empty circle indicates the start of the code segment
- Full circle indicates the end of the code segment

Selection Structures, ctd.

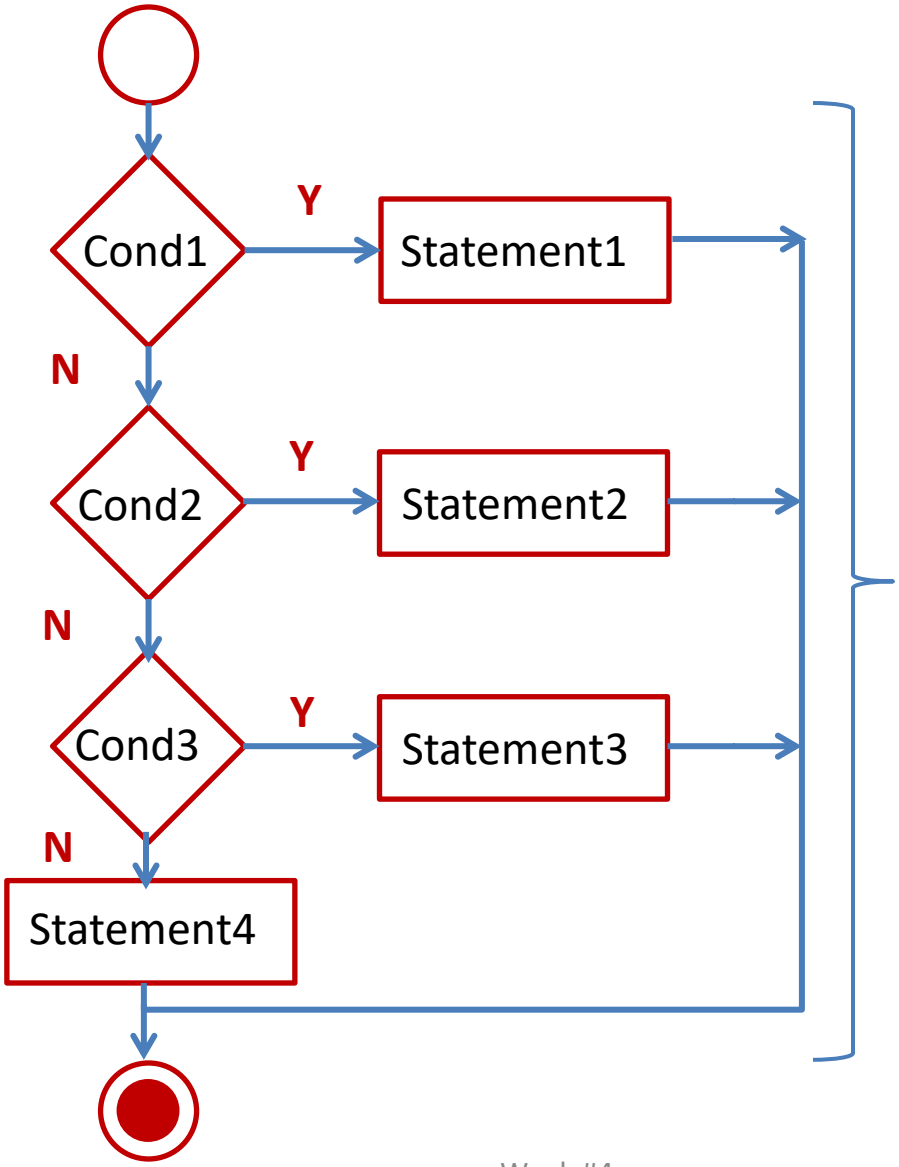
Flow-chart



If - else statements

Selection Structures, ctd.

Flow-chart



If – elseif – else statements

Selection Structures, ctd.

Are they all the same?

```
if(grade>90)
   disp('A');
elseif (grade>80)
   disp('B');
elseif (grade>70)
   disp('C');
else
   disp('F');
end
```

```
if(grade>90) %if_1
   disp('A');
else
   if (grade>80); %if_2
      disp('B');
   else
      if (grade>70) % if_3
         disp('C');
      else
         disp('F');
      end %end of if_3
   end %end of if_2
end % end of if_1
```

```
if(grade>90)
   disp('A');
   if (grade>80)
      disp('B');
      if (grade>70)
         disp('C');
      else
         disp('F');
      end
   end
end
```

switch-case Structure

• The switch-case structure is often used when a series of programming path options exists for a given variable, depending on its value.

• The switch-case is similar to the if-elseif-else.

• As a matter of fact, anything you can do with switch-case could be done with if-elseif-else.

• However, the code is a bit easier to read with switch-case, a structure that allows you to choose between multiple outcomes, based on some criterion.

• This is an important distinction between switch-case and if-elseif-else structure.

• The criterion can be either a scalar (a number) or a string. In practice, it is used more with strings than with numbers.

switch-case Structure, ctd.

```
switch ( variable )
case value1
   statements;
case value2
   statements;
otherwise
   statements;
end
```

The switch case in C programmer, you may have used switch/case in that language. One important difference in MATLAB ® is that once a “true” case has been found, the program does not check the other cases.

- **value1: constant integral expression** (i.e., anything that evaluates to a constant integer value) that the variable or expression on which the switch is based may assume.
- If no match occurs between the controlling expression's value and a case label, the default case executes

switch-case Structure, ctd.

```
city = input('Enter the name of a city in single       quotes:‘)
switch (city)
  case 'Boston'
    disp('$345')
  case 'Denver'
    disp('$150')
  case 'Honolulu'
    disp('Stay home and study')
  otherwise
    disp('Not on file')
end
```

If, when you run this script, you reply 'Boston' at the prompt, MATLAB responds

```
city =
Boston
$345
```