Lecture 5

Lecture 5: Selection Statements

- **Selection Statements**
  - Used to make choices as to whether a statement or a group of statements is executed or not, and how to choose between or among them.
- **Two Basic Statements**
  - If statement
  - Switch statement
- **Relational Expressions**
  - Conditions in if statement use expressions that are conceptually, or logically either true or false. These expressions are called relational expressions.
  - These expressions can use both relational and logical operators

Relational operators

>	greater than
<	less than
>=	greater than or equals
<=	less than or equals
==	equality
~=	inequility

Logical operators

||	or
&&	and
~	not

All logical operators operate on logical variables.

What's the difference between a=5 and a==5?

Lecture 5: Selection Statements

- **Relational Expressions (Cont)**
  - In MATLAB, logical true is represented by integer 1, and logical false is represented by integer 0. So, the expression 3 < 5 actually has the value 1.

Examples:

```
>> 3 < 5
ans =
     1
>> 9 < 2
ans =
     0
```

The value of a relational expression is either logical true or logical false.

Mathematics can be done on the logical values: >> 2+(3<5)

Questions: what's the meaning of
>> a=3<5

In the workspace window, the value shown for the result of these expressions would be true or false. The type of the result is logical.

- Comparing characters, for example 'a'<'c', is also possible. Characters are compared using their ASCII equivalent values. (See 'character encoding in Lecture 1)

Lecture 5: Selection Statements

- **Relational Expressions (Cont)**
  - In addition to these logical operators, MATLAB also has a function **xor**, which is the exclusive or function. It returns logical true if one (and only one) of the arguments is true.
  - Truth Table for Logical Operators

x	y	~x	x || y	x && y	xor(x,y)
True	True	False	True	True	False
True	False	False	True	false	True
false	false	True	false	false	false

- x and y are logical variable

Questions: can expression >> 0.1 < a && a < 0.3 be written as >> 0.1 < a < 0.3 ? Why?

Lecture 5: Selection Statements

- **The IF Statement**
  - The **if** statement chooses whether or not another statement, or a group of statements, is executed. The general form of the if statement is

```
if  condition
     action
end
```

The condition is a relational expression that is conceptually, or logically, either true or false. The action is a statement, or a group of statements, that will be executed if the condition is true.

Examples

```
>> num = -4;
>> if num < 0
    num = abs(num)
    end
num =
       4
```

Lecture 5: Selection Statements

➢The IF Statement

- In MATLAB, the concept of false is represented by the integer value of 0, but the concept of true can be represented by any nonzero value (not just the integer 1)

```
>> if  5
     disp('Yes, this is true!')
   end
Yes, this is true!
```

Here 5 represents logical true

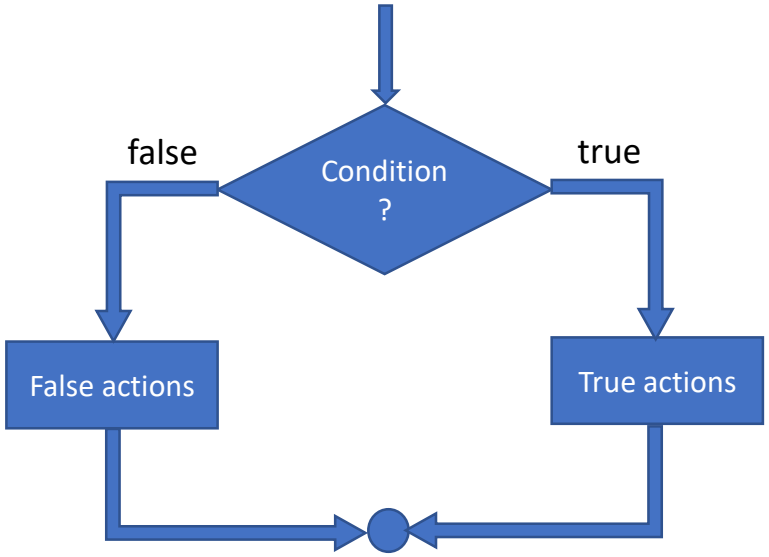
When we have a relational expression, and it is logically true, then that expression has a value of 1. However, logical true by be represented by any nonzero value.

Lecture 5: Selection Statements

- **The IF-ELSE Statement**
  - The if-else statement is used to choose between two statements, or two sets of statements. The general form is

```
if  condition
    action1
else
    action2
end
```

One of these actions, and only one, will be executed – which one depends on the value of the condition




```
>> var = 0.6;
>> if var < 0.5
disp('it was less than 0.5!')
else
disp('it was not less than 0.5!')
End
it was not less than 0.5!
```

Lecture 5: Selection Statements

- **Nested IF-ELSE Statement**
  - The if-else statement is used to choose between two statements. In order to choose from more than two statements, the if-else statements can be nested, one inside of another. Consider the mathematical function y=f(x)

    y = 1 for x < -1

    $y = x^2$ for $-1 \leq x \leq 2$

    y = 4 for x > 2

    Choosing which range could be accomplished with three separate if statements as follows

```
if  x < -1
    y = 1
end
if x >= -1 && x <= 2
    y = x^2
end
if x > 2
    y = 4
end
```

    this is not very efficient code: all three logical expressions must be evaluated, regardless of the range in which x falls.

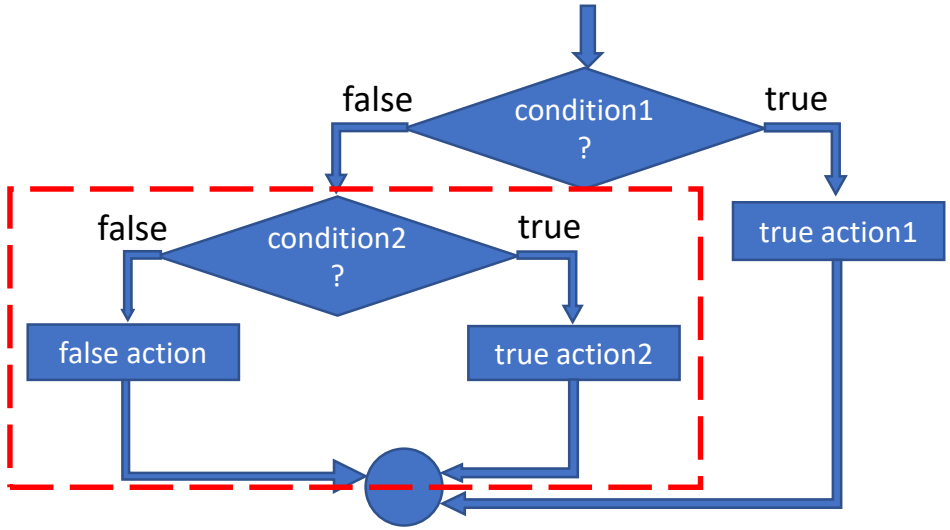
Lecture 5: Selection Statements

➢ Nested IF-ELSE Statement (cont)

- By using a nested if-else statement to choose among the three possibilities, not all conditions must be tested as they were in the previous example

```
if  x < -1
    y = 1
else
    if x >= -1 && x <= 2
        y = x^2
    else
        y = 4
    end
end
```

A nested if-else statement is used as the 2nd action of the outer if-else statement.



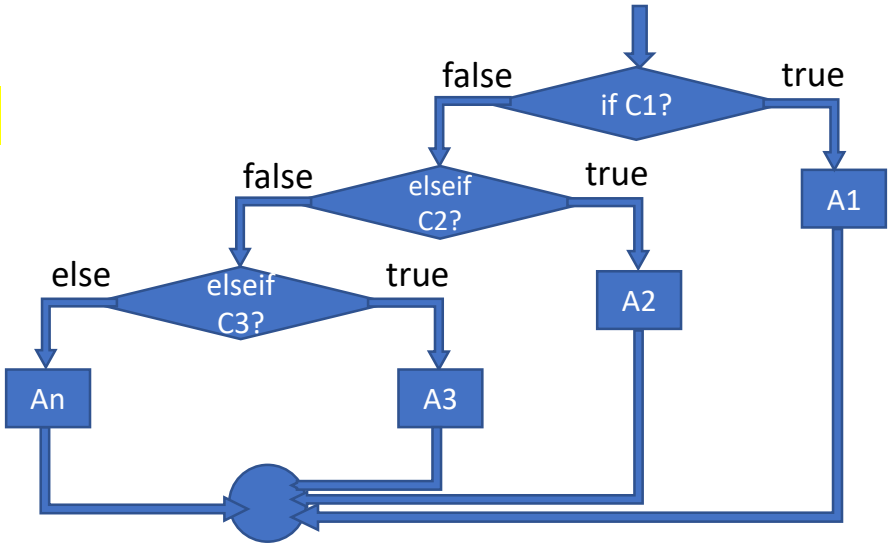
This is actually an example of a particular kind of nested if-else called a cascading if-else statement. In this type of nested if-else statement, the conditions and actions cascade in a stair-like pattern.

Lecture 5: Selection Statements

- **The Elseif Clause**
  - To choose from among more than two actions, MATLAB has another method of accomplishing this, using the elseif clause. The general form is

```
if  condition1
      action1
elseif condition2
      action2
elseif condition3
      action3
% etc: there can be many of these
else
      action    % the nth action
end
```

.



Lecture 5: Selection Statements

- **There are three ways to choose among more than two actions:**

1) using several separate if statements. least efficient
2) using nested if-else statements,
3) using an if statement with elseif clauses. Simplest and most efficient.

Lecture 5: Selection Statements

- Example: write a function to convert the integer grade of a quiz to the letter grade.

letgrade.m

```
function grade = letgrade(quiz)
% This function returns the letter grade corresponding
% to the integer quiz grade argument

% First, error-check
if quiz < 0 || quiz > 10
   grade = 'X';
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
```

- grade is the output argument of type char;
- letgrade is the function name
- quiz is the input argument of type integer

```
>> grade = letgrade(8)
grade =
      'B'
```

Lecture 5: Selection Statements

- **The Switch Statement**
  - A switch statement can often be used in place of a nested if-else or an if statement with many elseif clauses. Switch statements are used when an expression is tested to see whether it is **equal** to one of several possible values. The general form of the switch statement is:

```
switch switch_expression
    case caseexp1
        action1
    case caseexp2
        action2
    case caseexp3
        action3
    % etc: there can be many of these
    otherwise
        action_n
end
```

'otherwise' is used here, not 'else' as in the if-else statement.

Lecture 5: Selection Statements

- **The Switch Statement**
  - Example: write a function to convert the integer grade of a quiz to the letter grade.

switchletgrade.m

```
function grade = switchletgrade(quiz)
% This function returns the letter grade corresponding
% to the integer quiz grade argument using switch

if quiz < 0 || quiz > 10
   grade = 'X';
else
   switch quiz                  % quiz is the switch expression
      case 10                   % 10 is the case expression
         grade = 'A';           % action 1
      case 9
         grade = 'A';
      case 8
         grade = 'B';
      case 7
         grade = 'C';
      case 6
         grade = 'D';
      otherwise
         grade = 'F';
   end
end
```

If-else statement is used to check if the integer grade is in the reasonable range.

Generally the switch expression and case expression are not relation expressions. They are not logical type.

```
>> grade = switchletgrade(8)
grade =
      'B'
```

Lecture 5: Selection Statements

➢**Summary**

- If statement
- If-else statement
- Nested if-else statement
- Elseif clause
- Switch statement

- Homework on Canvas (Due next Wednesday)