PROGRAMMING WITH MATLAB

WEEK 5

CONTROL STRUCTURES

CONTROL STRUCTURES

- Algorithms are sequences of explicitly defined instructions to solve a particular problem.
- The instructions are ordered and can be numbered. An algorithm, however, should be able to change the order of instructions by using a control structure.
- Sequential operations: Instructions executed in order.
- Conditional operations: Control structures select the appropriate ones from the instructions based on whether a certain condition is met.
- Iterative operations: Control structures execute a group of instructions for a certain number of times or as long as certain conditions are met

RELATIONAL OPERATORS

operator	description
==	Equal to
~=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

RELATIONAL OPERATORS

```
>> x = -3;
>> y = 5.2;
>> z = x<y
z =
  logical
   1
>> x = [-3, 3, 5];
>> y = [5.2, 2.5, 3];
>> z = x<y
z =
  1×3 logical array
   1   0   0
>> z = x(x>y)    % (finds all the elements in x that are greater than the corresponding elements in y)
z =
     3     5
```

LOGICAL OPERATORS

operator	description
&	AND (x&y)
|	OR
~	NOT
all	All true
any	Any one true
&&	AND (short-circuit AND), scalar
||	OR (short-cirduit OR), scalar
xor	XOR

LOGICAL OPERATORS

operator	description
isempty(X)	Returns a 1 if X is an empty matrix and 0 otherwise
isinf(X)	Returns an array of the same dimension as X, with ones where X has ‘inf’ and zeros elsewhere
isnan(X)	Returns an array of the same dimension as X with ones where X has ‘NaN’ and zeros elsewhere
ischarX()	Returns a 1 if X is a character array and 0 otherwise
isnumeric(X)	Returns a 1 if X is a numeric array and 0 otherwise
isreal(X)	Returns a 1 if X has no elements with imaginary parts and 0 otherwise

ORDER OF PRECEDENCE FOR OPERATORS

- First: Parentheses (starting with the innermost pair)
- Second: Arithmetic operator and logical NOT (left to right)
- Third: Relational operators (left to right)
- Fourth: Logical AND
- Fifth: Logical OR

IF/ELSE/ELSEIF

- IF : Basic flow control in all programming languages

Syntax:

```
if logical expression/condition
        statements
end
```

- ELSE

Syntax:

```
if logical expression
      statement group 1
else
      statement group 2
end
```

- ELSEIF

Syntax:

```
if logical expression 1
   statement group 1
elseif logical expression 2
   statement group 2
else
   statement group 3
End
```

Parentheses are not needed, and command blocks are between reserved words (such as if, end)

IF/ELSE/ELSEIF

You can nest any number of if statements. Each if statement requires an end keyword.

Avoid adding a space after else within the elseif keyword (else if). The space creates a nested if statement that requires its own end keyword.

```
x = 5;
min = 1;
max = 10;

if (x >= min) && (x <= max)
    disp('Value within specified range.')
elseif (x > max)
    disp('Value exceeds maximum value.')
else
    disp('Value is below minimum value.')
end
```

LOOPS

- while loops: Similar to a more general for loop. No need to know the number of iterations

Syntax:

```
while conditional expression
        statements
end
```

```
>> x = 1;
while x < 13
       x = x + 3;
end
```

The command block is executed as long as the conditional expression is correct

LOOPS

The infinite loop must be avoided!

If you inadvertently create an infinite loop (that is, a loop that never ends on its own), stop execution of the loop by pressing Ctrl+C.

To programmatically exit the loop, use a break statement. To skip the rest of the instructions in the loop and begin the next iteration, use a continue statement.

When nesting a number of while statements, each while statement requires an end keyword

LOOPS

- The switch structure: The switch structure provides an alternative to using the if, elseif, and else commands

Syntax:

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

LOOPS

```
x = input('Enter a number: ');

switch x
    case -1
        disp('negative one')
    case 0
        disp('zero')
    case 1
        disp('positive one')
    otherwise
        disp('other value')
end
```