Introduction to Mathematical Software and Programming

Session 3

(w/c 3 March 2025)

Logical statement in MATLAB

In MATLAB, logical (Boolean) variables/statements take value 1 (true) or 0 (false).

We can create logical statements using combination of arithmetic, relational and logical operators. *(Sem-1, CELEN086 Lecture 1)*

Operator	Description
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
~=	Not equal to
&	AND operator
|	OR operator
~	NOT operator

A	B	A && B	A || B
1	1	1	1
1	0	0	1
0	1	0	1
0	0	0	0

A	~A
1	0
0	1

Note:
In MATLAB on this module, we will stick to the MATLAB syntax && for logic AND, || for logic OR.

Precedence of operators

Predict the outputs and try the following statements in MATLAB.

```
x=3; y=4; z=5; u=true; v = [ ];

~(x<z)

x>y||y<z

x^2+y^2==z^2 && x+y~=z

isempty(v)

w = ~(x<z)|| ~u

x~=y || isprime(5) && isempty([2,3])
```

Precedence	Operator
1	Parentheses ()
2	Transpose (.′), power (.^), matrix power (^)
3	Unary plus (+), unary minus (−), logical negation (~)
4	Multiplication (.∗), right division (./), left division (.\), matrix multiplication (∗), matrix right division (/), matrix left division (\)
5	Addition (+), subtraction (−)
6	Colon operator (:)
7	Less than (<), less than or equal to (≤), greater (>), greater than or equal to (≥), equal to (==), not equal to (~=)
8	Element-wise AND, (&)
9	Element-wise OR, (|)

Note:
isprime() and isempty() are MATLAB built-in logical functions.

[ ] represents an empty array.

IF structure

Logical statements can be used as "conditions" in the IF structures.

(Sem-1, CELEN086 Lecture 1&2)

gradeConverter1.m

```
1     % US letter grade convertor
2     % using single IF structure
3 -   clear;clc
4 -   grade = input("Enter the grade:\n");
5 -   if grade>=90
6 -       disp("A")
7 -   elseif grade>=80
8 -       disp("B")
9 -   elseif grade>=70
10 -      disp("C")
11 -  elseif grade>=60
12 -      disp("D")
13 -  else
14 -      disp("Fail")
15 -  end
```

gradeConverter2.m

```
1     % US letter grade convertor
2     % using Nested-IF structure
3 -   clear;clc
4 -   grade = input('Enter the grade:\n');
5 -   if grade<60
6 -       disp('Fail')
7 -   else
8 -       if grade>=90
9 -           disp('A')
10 -      elseif grade>=80
11 -          disp('B')
12 -      elseif grade>=70
13 -          disp('C')
14 -      else
15 -          disp('D')
16 -      end
17 -  end
```

Iteration in Programming

Iteration in programming refers to the process of repeatedly executing a set of instructions or code until a specific condition is met. This is commonly achieved using loops, such as for loops and while loops.

- For loop
  - Count-controlled iteration
  - Iterate a fixed number of times and stop
- While loop
  - Condition-controlled iteration
  - Iterate while certain condition is met and stop otherwise

For loop

```
for index = vector
    statements
end
```

Note:

- *index variable repeatedly takes values from a vector a:increament:b*
- *The repeated statements contains the index variable.*
- *The statements will be repeated k times, k=length(vector)*

```
for i = 1:1:10
    disp(i)
end
```

```
x = 0;
for i = 1:10
    x = x+i;
end
disp(x)
```

```
y = 1;
for i = 1:10
    y = y*i;
end
disp(y)
```

Example: Compute the sum of all odd numbers from 1 to 10.

```
x = 0;
for n = 1:2:9
    x = x+n;
end
disp(x)
```

Odd numbers can be manipulated either in the index vector or in the statement.

```
x = 0;
for n = 1:5
    x = x+(2*n-1);
end
disp(x)
```

Sample code (Fibonacci sequence)

Write a script for displaying the first 10 Fibonacci numbers (1,1,2,3,5,8,…)

fibo.m

```
% Display First n Fibonacci numbers
n = 10;
F = zeros(1,n); % preallocation for the array
F(1)=1; F(2)=1;
for i=3:n
    F(i) = F(i-1)+F(i-2); % updating array values
end
fprintf('First %d Fibonacci numbers are:\n',n)
disp(F)
```

F is initialized as a zero array of length n.

Then we update its elements by the iterative formula using For Loop

Command Window

```
>> fibo
First 10 Fibonacci numbers are:
     1     1     2     3     5     8    13    21    34    55
```

Nested For Loop

Create a 10×10 matrix A with element values 1 to 100 in an increasing order. For example, 1 to 10 in the first row, 11 to 20 in the second row... and so on. Then use appropriate statements to replace elements that are not prime numbers by 0.

Analysis:

- We need a 10×10 array (matrix)
- The element in the i-th row and j-th column is indexed by $A(i, j)$

Sample code

```
1 -    clear;clc
2 -    n = 10;
3 -    A = zeros(n,n); % preallocation
4 -    for i = 1:n % iterating rows
5 -        for j = 1:n % iterating columns
6 -            A(i,j) = n*(i-1)+j; % update values in A
7 -        end
8 -    end
9 -    disp(A) % display original array A
10 -   index = ~isprime(A); % logical array for non-prime number
11 -   A(index) = 0; % update all non-prime numbers as 0
12 -   disp(A) % display the final array A
```

While Loop

```
while expression
    statements
end
```

Note:

- *Normally, the expression is a conditional statement*
- *The statements will be repeated when the conditional expression is true.*
- *The statements contains updates that affect the conditional expression (from true to false to end the while loop after repeating certain times)*

Example: Compute the sum of all integers from 1 to 100.

```
% while loop
mySum = 0;
n = 1;
while n<=100
    mySum = n+mySum;
    n = n+1;
end
disp(mySum)
```

These two statements will be repeated executed until n exceeds 100.

Solve the same question using for loop:

```
% for loop
mySum = 0;
for n = 1:100
    mySum = n+mySum;
end
disp(mySum)
```

Sample code

Determine the largest possible number n such that

$$\sum_{k=1}^{n} 2^k = 2^1 + 2^2 + 2^3 + \cdots + 2^n \leq 1000$$

Analysis (algorithm):

- Step 1: Initialize a variable mySum for storing the cumulative sum
- Step 2: (Iteration) Keep adding the term 2^k to mySum for $k = 1,2, \ldots$ repeatly.
- Step 3: We don't know how many iterations are needed in Step 2, so While Loop can be used instead of For Loop, for checking if mySum exceeds 1000.

While loop (method 1)

```
1 -    clear;clc
2 -    mySum = 0;
3 -    k = 0;
4 -    while mySum<=1000
5 -        k = k+1;
6 -        mySum = mySum+2^k;
7 -    end
8 -    disp(k-1)
```

You have multiple ways of initialize the counter variable k (line 3), and arrange those two statements (line 5,6).

Analyze the code segments to see how they are logically related (including line 8).

While loop (method 2)

```
1 -    clear;clc
2 -    mySum = 0;
3 -    k = 1;
4 -    while mySum<=1000
5 -        mySum = mySum+2^k;
6 -        k = k+1;
7 -    end
8 -    disp(k-2)
```

break command

In For/While Loops, we may have the need of quitting the iteration process earlier before the predefined conditions.

```
>> help break
 break Terminate execution of WHILE or FOR loop.
    break terminates the execution of FOR and WHILE loops.
    In nested loops, break exits from the innermost loop only.

    break is not defined outside of a FOR or WHILE loop.
```

You can find one application of using "break" in linear search (Question 10, Lab Worksheet 3)

Self-study

- Check Session 2 Solution Set available on Moodle
- Complete lab worksheet 3
- Complete homework exercise sheet 3

For environmental science and/or transferred students: review the following knowledge points from Sem-1 CELEN086 (Session 1 to Session 4) or from Internet resources.

- *Boolean(logical) statements*
- *IF and Nested IF structures*
- *Prime numbers and Fibonacci sequence*