

MATLAB 程式撰寫

- 程式與函數之編寫
- 關係與邏輯運算元
- 迴圈與程式流程控制



MATLAB 之程式類型 (m檔案)

- 一個用來執行MATLAB的最普遍方式，就是在命令視窗中一次輸入一個指令。m 檔提供另一條進行運算的途徑，可以擴展MATLAB解決問題的能力。一個**m**檔 (m-file) 包括一系列可以在同一時間執行的敘述。
- M檔有兩種形式，副檔名皆為m (filename.m)
 - ☐ 腳本檔案 (script file)
 - ☐ 函數檔案 (function file)



腳本檔案

- 所謂腳本檔案 (script file) ，是指一系列儲存於檔案中的MATLAB指令。
- 腳本檔案最常被用來保留一連串的命令，做重複的使用。
- 這些腳本可以在命令視窗中輸入檔案名稱來執行，或者可以利用編輯視窗中的選單，下拉後選取**Debug**及**Run**執行。即逐一執行檔案內的指令。

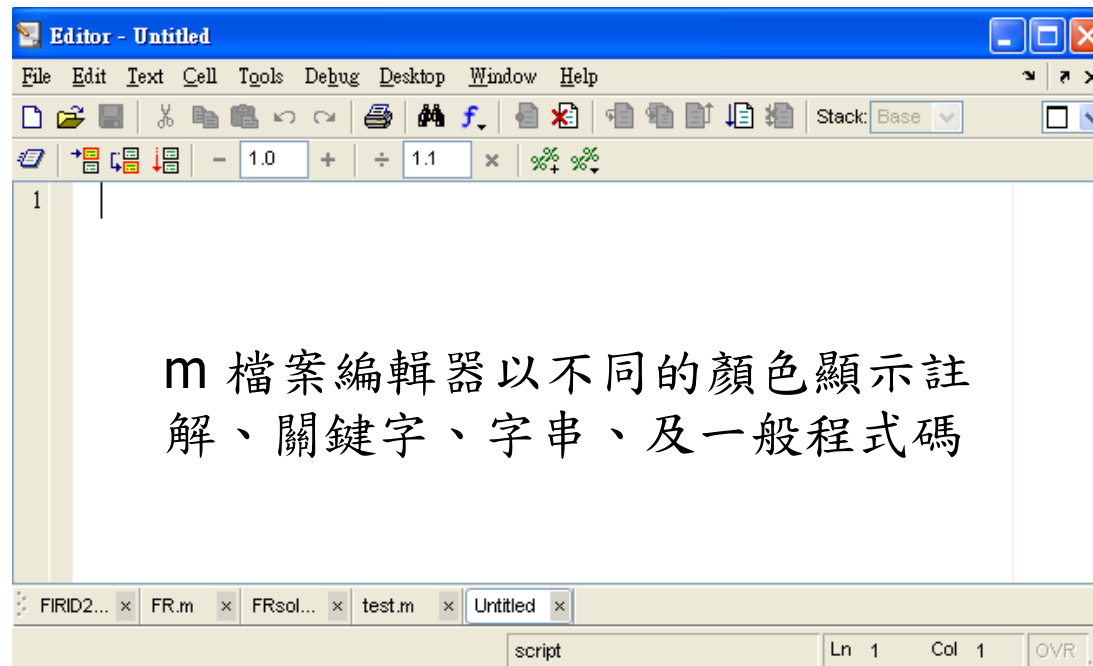


腳本 (Script)

- 若在目前目錄下有一個m檔案“test.m”，可用 type 指令顯示其內容：
 >> **type test.m**
- 執行底稿所產生的變數都存放在 MATLAB 的工作空間 (Workspace)
- 優點
 - 適用於簡單但重複性高的程式碼
 - 產生的變數保留在基本工作空間中
 - 變數檢視及除錯容易
- 缺點
 - 不支援輸入及輸出引數 (Input/Output Arguments)
 - 產生的變數保留在基本工作空間中
 - 變數互相覆蓋而造成程式錯誤

m 檔案編輯器

- m 檔案是文字檔
 - 可以用各種文字編輯器修改
- MATLAB 提供了內建的「m 檔案編輯器」(m-file editor)
 - 點選指令視窗的 file/open 下拉式選單，開啟 m 檔案編輯器
 - 或在指令視窗直接鍵入「edit filename.m」或「open filename.m」





函數檔案

- 所謂函數檔案 (function file) 就是以 **function** 這個文字起頭的m檔。
- 和腳本檔案不同，函數檔案可以接受引數並且傳回輸出值。即可接受**輸入變數**，並將結果送至**輸出變數**。
- 函數適用於大型程式碼
 - 使程式碼模組化 (Modularized) 並易於維護與改進

函數檔案語法

- `function outvar = funcname(arglist)`
 % help comments
 statements
 outvar = value;

outvar = 表示輸出變數的名稱

funcname = 表示函數的名稱

arglist = 表示函數的引數清單（也就是以逗點界定的值，可以用來傳遞到函數）

help comments = 表示可以提供使用者有關此函數資訊的文字（可以在命令視窗中鍵入 `help funcname` 來啟動）

statements = 表示可以用來計算 *value* 並且指派給 *outvar* 的 MATLAB 敘述。



函數 (Function)

- 用 type 指令顯示其內容：

- `>> type func1.m`

```
function average = func1(vector)
```

```
average = sum(vector)/length(vector);    % 計算平均值
```

- 第一列為函數定義列 (Function Definition Line)

- 定義函數名稱 (func1, 最好和檔案的檔名相同)
 - 輸入引數 (vector)
 - 輸出引數 (average)
 - function 為關鍵字

- 第二列以後為函數主體 (Function Body)

- 規範函數運算過程，並指定輸出引數的值

函數的呼叫

- 呼叫的基本語法 (一個函數可以有多輸入及輸出)

```
[Output1, Output2, ...] = funcname(input1, input2, ...)
```

- func1之呼叫方式

```
>> vec = [1 5 3];  
>> ave = func1(vec)  
ave =  
    3
```

- func2.m 可接受兩個輸入並產生兩個輸出

```
function [ave1, ave2] = func2(vector1, vector2);  
ave1 = sum(vector1)/length(vector1);  
ave2 = sum(vector2)/length(vector2);
```

- func2.m 的呼叫方式

```
>> [a, b] = func2([1 2 3], [4 5 6 7 8])  
a =  
    2  
b =  
    6
```



函數命名的限制

- 函數名稱和變數名稱有相同的限制
 - 只接受前 31 個字母（MATLAB 5.x）或前 63 個字母（MATLAB 6.x 和 7.x）
 - 以英文字母作為開頭

- 函數名稱和檔案名稱不同
 - 仍可依檔案名稱呼叫檔案
 - 函數名稱將被忽略（以檔名呼叫）



次函數

- 一個 m 檔案可以包含一個以上的函數
 - 第一個函數稱為主函數 (Primary Function)
 - 其他則為次函數 (Subfunctions)
 - 次函數只能被同檔案中的函數 (主函數或次函數) 呼叫，但不可被不同檔案的其他函數呼叫
- 主函數與次函數的位置
 - 主函數必需出現在最上方
 - 其後接上任意數目的次函數
 - 次函數的次序並無任何限制



主函數與次函數範例

- func3.m 包含一個主函數及一個次函數
- 次函數的功能是計算倒數向量

```
function out = func3(x)           % 主函數
    recip = reciproc(x);
    out = sum(recip);
```

```
% Definition for subfunctions
function output = reciproc(input) % 次函數
    output = 1./input;
```

- 呼叫此函數

```
>> ans = func3([1 2 3])
ans =
    1.8333
```

Inline Function

- 對於簡單的數學函數，可用inline指令

Ex.

- ☐ `f = inline('x^2')`
- ☐ `g = inline('sin(2*pi*f + theta)')`
- ☐ `g = inline('sin(2*pi*f + theta)', 'f', 'theta')`

```
>> f=inline('x^2+log(x)')
```

```
>> f(2)
```

```
ans =
```

```
4.6931
```

argument





匿名函數

- 匿名函數 (anonymous function) 允許你在不建立m檔的情況下，建立一個簡單的函數，我們可以以下列語法在命令視窗中定義這些函數：

fhandle = @(*arglist*) *expression*

fhandle = 函數的名稱 (function handle)

arglist = 以逗號分開傳至函數的輸入參數

expression = 是一個MATLAB的正確表示式



函數中的函數

- 「函數中的函數」是一個將函數當作輸入參數以操作另一函數的函數，此函數可傳遞「函數中的函數」作為傳遞函數 (passed function)。



區域變數與全域變數

■ 區域變數（Local Variables）

- 每一個函數在運算時，均佔用個別的記憶體
- 此工作空間和 MATLAB 的基本工作空間或是其他函數的工作空間是互相獨立的
- 不同空間的變數是完全獨立，不會相互影響
- 不同工作空間的變數，稱為「區域變數」

■ 全域變數（Global Variables）

- 減少變數的傳遞，可用「全域變數」（Global Variables）
- 使用全域變數前，需先進行變數宣告

全域變數範例

```
function func4
global X                % 全域變數宣告
X = X + 2;
fprintf('The value of X in "func4" is %g.\n', X);
```

- func4.m沒有輸出和輸入，只宣告全域變數 X，將 X 的值加 2，並印出其值

- 測試

```
>> global X    % 在基本工作空間進行全域變數 x 的宣告
>> X = 2;
>> fprintf('The value of X in the base workspace is %g.\n', X);
The value of X in the base workspace is 2.

>> func4;
The value of X in "func4" is 4.

>> fprintf('The value of X in the base workspace is %g.\n', X);
The value of X in the base workspace is 4.
```

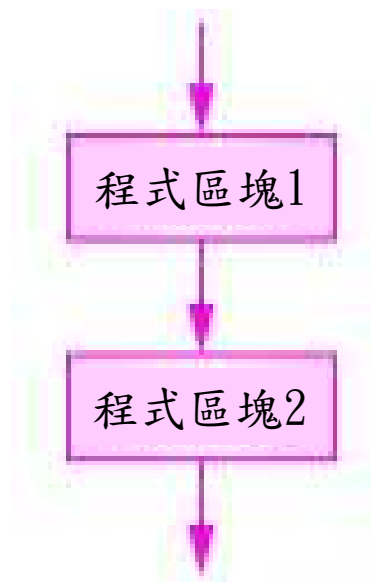


全域變數的使用原則

- 盡量少用全域變數
 - 全域變數使程式的流程不透明，造成程式除錯或維護的困難
- 使用全域變數，請遵循下列兩原則
 - 使用前一定要宣告
 - 使用全部大寫或較長的變數名稱，以資區別
- 檢視工作空間的變數，輸入whos global
- 清除所有工作空間的全域變數 X，需使用
clear global X

MATLAB 程式流程控制

循序結構





結構化的程式

- m檔會循序地執行每一個指令，也就是從函數檔案的最上面一行開始，逐行執行程式的敘述直到最後一行。因為僵化的順序會限制住功能，所有的電腦程式語言一定都包括讓程式使用非循序途徑的敘述（指令）。這些可以分類成：
 - 迴圈 (Loop) [或稱反覆執行 (Repetition)] 。
流程的迴圈可以讓敘述反覆地被執行。
 - 決策 (Decisions) [或稱選擇 (Selection)] 。
根據決策決定流程的分支。

關係運算元

■ <

■ <=

■ >

■ >=

■ == equal to

■ ~= not equal to

■ 1真 0偽

範例	運算子	關係
x == 0	==	等於
unit ~= 'm'	~=	不等於
a < 0	<	小於
s > t	>	大於
3.9 <= a/3	<=	小於或等於
r >= 0	>=	大於或等於



Ex:

```
>>a=[1 2 3 4];
```

```
>>b=[5 6 7 2];
```

```
>>a>b
```

```
ans =
```

```
0    0    0    1
```

```
>>tf=b-(a>2)
```

```
tf =
```

```
5    6    6    1
```

Ex:

```
>>2>3
```

```
ans =
```

```
0
```

```
>>2<=3
```

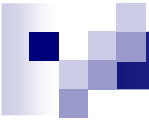
```
ans =
```

```
1
```

```
>>2==3
```

```
ans =
```

```
0
```



Ex: Find out $\frac{\sin(x)}{x}$, $x=-1:1/3:1$

```
>>x=-1:1/3:1;
```

```
>>sin(x)./x
```

Warning: Divide by zero

```
ans =
```

```
0.8415 0.9276 0.9816 NaN 0.9816 0.9276 0.8415
```

```
>>x==0
```

```
ans =
```

```
0 0 0 1 0 0 0
```

```
>>x=x+(x==0)*eps; NOTE: eps 内定精確度 2.2204e-16
```

```
>>sin(x)./x
```

```
ans =
```

```
0.8415 0.9276 0.9816 1.0000 0.9816 0.9276 0.8415
```

邏輯運算元

- **& (And)** (以及)。用來將兩個表示式做邏輯連接 (logical conjunction)。

$expression_1 \& expression_2$

如果兩個 $expression$ 都為真，則結果為真。任何一個或者兩個 $expression$ 為假，則結果為假。

- **| (Or)** (或者)。用來執行兩個表示式的不連接 (logical disjunction)。

$expression_1 \mid expression_2$

如果任何一個或者兩個 $expression$ 為真，則結果為真。

- **~ (Not)** (反相)。將表示式取邏輯上的負值。

$\sim expression$

如果 $expression$ 為真，則結果為假。相反地，如果 $expression$ 為假，則結果為真。

真值表

		最高	→		
x	y	~x	x & y		最低 x y
T	T	F	T		T
T	F	F	F		T
F	T	T	F		T
F	F	T	F		F

邏輯運算元

& and
| or
~ not

Ex:

```
>>a=1:9;
```

```
>>tf=a>4
```

```
tf =
```

```
0 0 0 0 1 1 1 1 1
```

```
>>tf=~(a>4)
```

```
tf =
```

```
1 1 1 1 0 0 0 0 0
```

```
>>tf=(a>2)&(a<6)
```

```
tf =
```

```
0 0 1 1 1 0 0 0 0
```



迴圈指令

- 和字面上的意義一樣，迴圈可以將一個運算不斷地重複。
- MATLAB 提供兩種迴圈指令，一種是 for 迴圈，另一種是 while 迴圈
 - **for迴圈** (for loop) 在進行**指定次數**的重複動作之後停止。
 - **while迴圈** (while loop) 則在**某一個邏輯條件成立**時終止。

for 迴圈指令

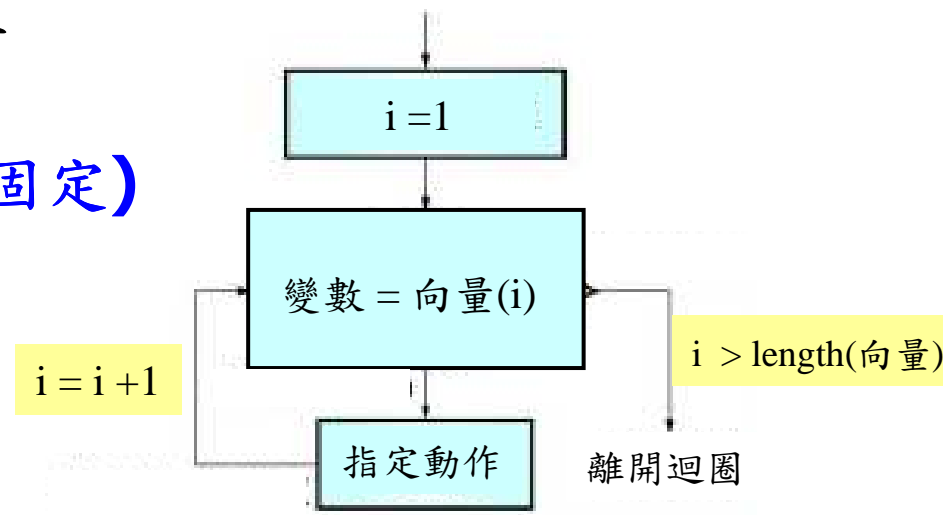
■ for...end 結構

一個for迴圈會重複執行敘述直到指定次數。

■ for 迴圈的使用語法如下

```
for 變數 = 向量 (迴圈數固定)
    運算式
end
```

- 其中變數的值會被依次設定為向量的每一個元素值，來執行介於 for 和 end 之間的運算式



for 迴圈範例

```
for i=1:3  
    y(i)=cos(i)  
end
```

執行結果

y =

0.5403

y =

0.5403 -0.4161

y =

0.5403 -0.4161 -0.9900

Ex: $1+2+3+4+5...+10=?$

```
sum=0;  
for i=1:10;  
    sum=sum+i;  
end  
fprintf('sum= %5.0f \n', sum)
```

執行結果

sum = 55

- 若要跳出 for 迴圈，可用 **break** 指令

巢狀式 for 迴圈

- for 迴圈可以是多層或巢狀式的

```
for 變數1 = 向量1  
    運算式1  
    for 變數2 = 向量2  
        運算式2  
    end  
    :  
    :  
end
```

- 巢狀結構 (nesting structure)：結構可以「築巢」於另外的結構之中。
- 使用內縮 (indentation) 使隱藏在框架之中的邏輯結構變得清晰。



向量化

- for迴圈是容易執行並且容易了解的。然而，對於MATLAB，根據某一指定次數重複執行一個敘述有時並非是最有效率的方式。因為MATLAB有能力直接對陣列做運算，向量化 (vectorization) 可以提供一個更有效率的選擇。

Loop迴圈	向量化
<pre>i = 0; for t = 0:0.02:50 i = i + 1; y(i) = cos(t); end</pre>	<pre>t = 0:0.02:50; y = cos(t);</pre>

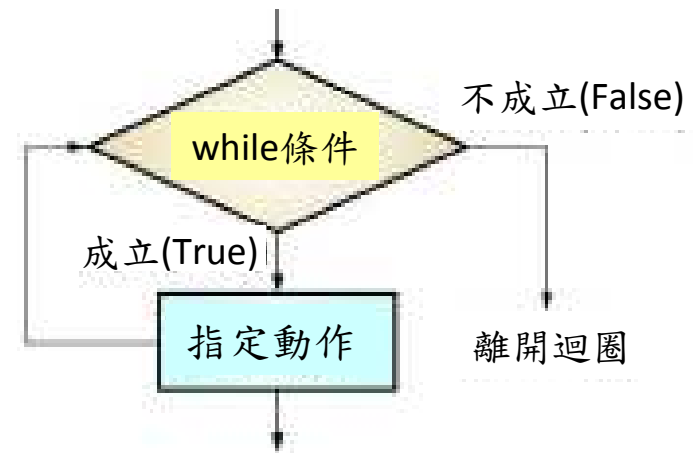
while 迴圈指令

■ while...end 結構

while迴圈只要某一邏輯條件仍為真，就會不停地重複執行。

■ while 迴圈使用語法如下：

```
while 條件式 (迴圈數未知)
    運算式
end
```



□ 若條件式成立則執行介於 while 和 end 之間的運算式

while 迴圈範例

Ex: $1+2+3+\dots+n > 50$ 最小之n值?

程式

```
sum=0;
n=0;
while sum<=50
    n=n+1;
    sum=sum+n;
end
fprintf('1+2+...+n >50 最小之n值= %3.0f \n', n)
```

執行結果

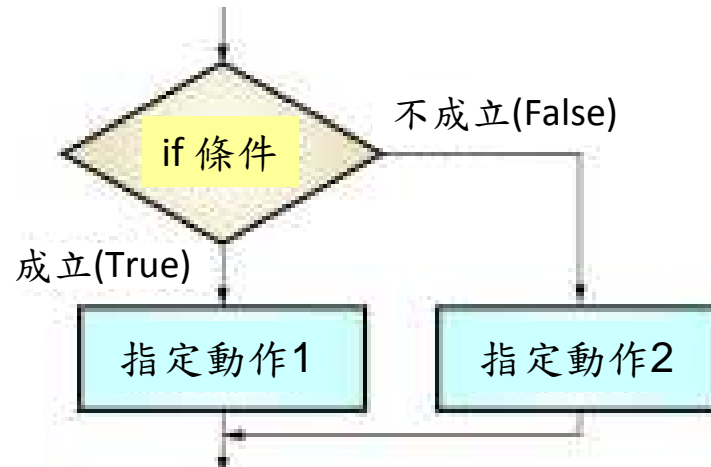
$1+2+\dots+n > 50$ 最小之n值= 10

- 若要跳出 while 迴圈，亦可用 **break** 指令

條件指令

- MATLAB 支援二種條件指令，一種是 **if-else-end** 條件指令，另一種是 MATLAB 在第五版之後開始支援的 **switch - case - otherwise** 條件指令
- 最常用的條件指令是 **if - else - end**，其使用語法為：

```
if 條件式  
    運算式一  
else  
    運算式二  
end
```



- 當條件式成立時，MATLAB 將執行運算式一，否則，就執行運算式二。若不需使用運算式二，則可直接省略 **else** 和運算式二

if - else - end 範例

- 根據向量 y 的元素值為奇數或偶數，來顯示不同的訊息：

```
y = [0 3 4 1 6];  
for i = 1:length(y)  
    if rem(y(i), 2)==0  
        fprintf('y(%d) = %d is even.\n', i, y(i));  
    else  
        fprintf('y(%d) = %d is odd.\n', i, y(i));  
    end  
end
```

```
y(1) = 0 is even.  
y(2) = 3 is odd.  
y(3) = 4 is even.  
y(4) = 1 is odd.  
y(5) = 6 is even.
```

- 上述的 if - then - else 為雙向條件，亦即程式只會執行「運算式一」或「運算式二」，不會有第三種可能

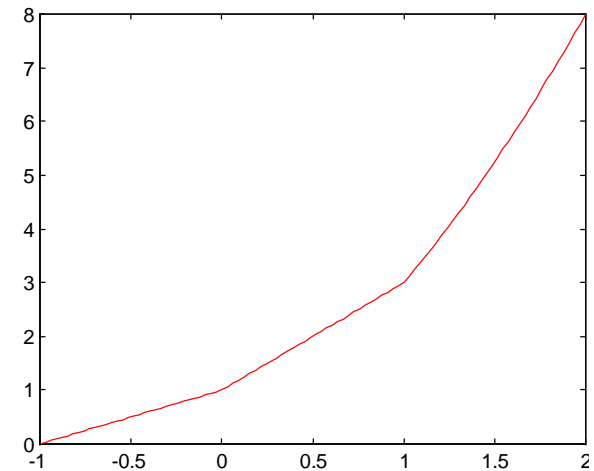
多向條件指令

- MATLAB 亦可執行多向條件，若要進行更多向的條件，只需一再重覆 **elseif** 即可

```
if 條件式一  
    運算式一  
elseif 條件式二  
    運算式二  
else  
    運算式三  
end
```

Ex. $f(x) = \begin{cases} x+1 & , x \leq 0 \\ 2x+1 & , 0 < x \leq 1 \\ x^2 + 2x & , 1 < x \leq 2 \end{cases}$ plot f(x) v.s. x

```
x=linspace(-1,2,100);  
for i=1:length(x)  
    if x(i)<=0  
        y(i)=x(i)+1;  
    elseif x(i)<=1  
        y(i)=2*x(i)+1;  
    else  
        y(i)=x(i)^2+2*x(i);  
    end  
end  
plot(x,y)
```



提前中斷

- 有一種特別的架構可以在邏輯條件為真的情況下，可隨時從迴圈的任何一處跳出。雖然MATLAB沒有這樣的結構，仍可以用一個特別版本的while迴圈來模擬這個功能。此版本的語法叫做`while...break`，使用如下：

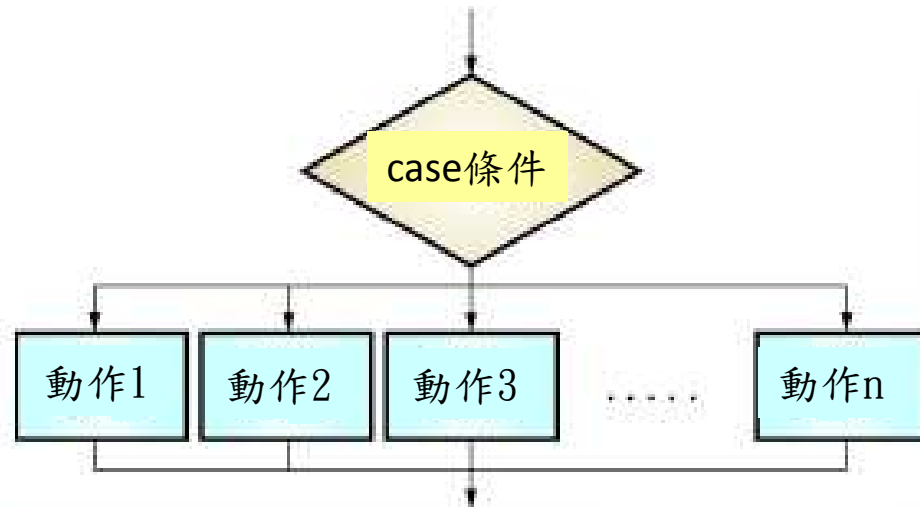
```
while (1)
    statements
    if condition, break, end
    statements
end
```

其中break終止迴圈的執行。因此，在此條件為真的情況下，單行if可以用來跳出這個迴圈。這樣的結構稱為中間測試迴圈 (midtest loop)。

switch-case-otherwise 指令

- MATLAB 在第五版開始支援 switch-case-otherwise 的多向條件指令，其使用語法如下：

```
switch expression
case value 1
    statement 1
case value 2
    statement 2
case value n-1
    statement n-1
otherwise
    statement n
end
```



- 在上述語法中，expression 為一數值或字串，當其值和 value k 相等時，MATLAB 即執行 statement k 並跳出 switch 指令。若 expression 不等於 value k， $k=1, 2, \dots, n-1$ ，則 MATLAB 會執行 statement n 並跳出 switch 指令。

switch-case-otherwise 範例

- 欲根據月份來判斷其季別，可輸入如下：

```
for month = 1:12
    switch month
        case {3,4,5}
            season = 'Spring';
        case {6,7,8}
            season = 'Summer';
        case {9,10,11}
            season = 'Autumn';
        case {12,1,2}
            season = 'Winter';
    end
    fprintf('Month %g ==> %s.\n', month, season);
end
```

Month 1 ==> Winter.

⋮

Month 12 ==> Winter.

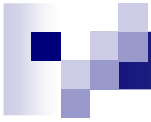


Exercise

- $x=[1:10]$ ， $f(x)=\sin(x)$ ，請分別利用 for 迴圈與 while 迴圈計算這些 x 點的函數值，以一個向量表示之。
- 若 $x=[0:0.5:5]$ ，請重做上題。
- 完成一個 5×4 的矩陣，其中各元素之值為該元素行索引值加上列索引值的和。
- 寫一個 MATLAB 的遞迴函數 `fibo.m` 來計算 Fibonacci 數列，其定義如下：
$$\text{fibo}(n+2) = \text{fibo}(n+1) + \text{fibo}(n)$$

此數列的啟始條件如下：

$$\text{fibo}(1) = 0, \text{fibo}(2) = 1$$



```
function out = fibo(n)
```

```
    if n==1
```

```
        out=0;
```

```
        return;
```

```
    elseif n==2
```

```
        out=1;
```

```
        return;
```

```
    else
```

```
        out=fibo(n-1)+fibo(n-2);
```

```
    end
```