Chapter 4
分支宣告與程式設計

Hung-Yuan Fan (范洪源)

Department of Mathematics,
National Taiwan Normal University, Taiwan

Spring 2017

Outline

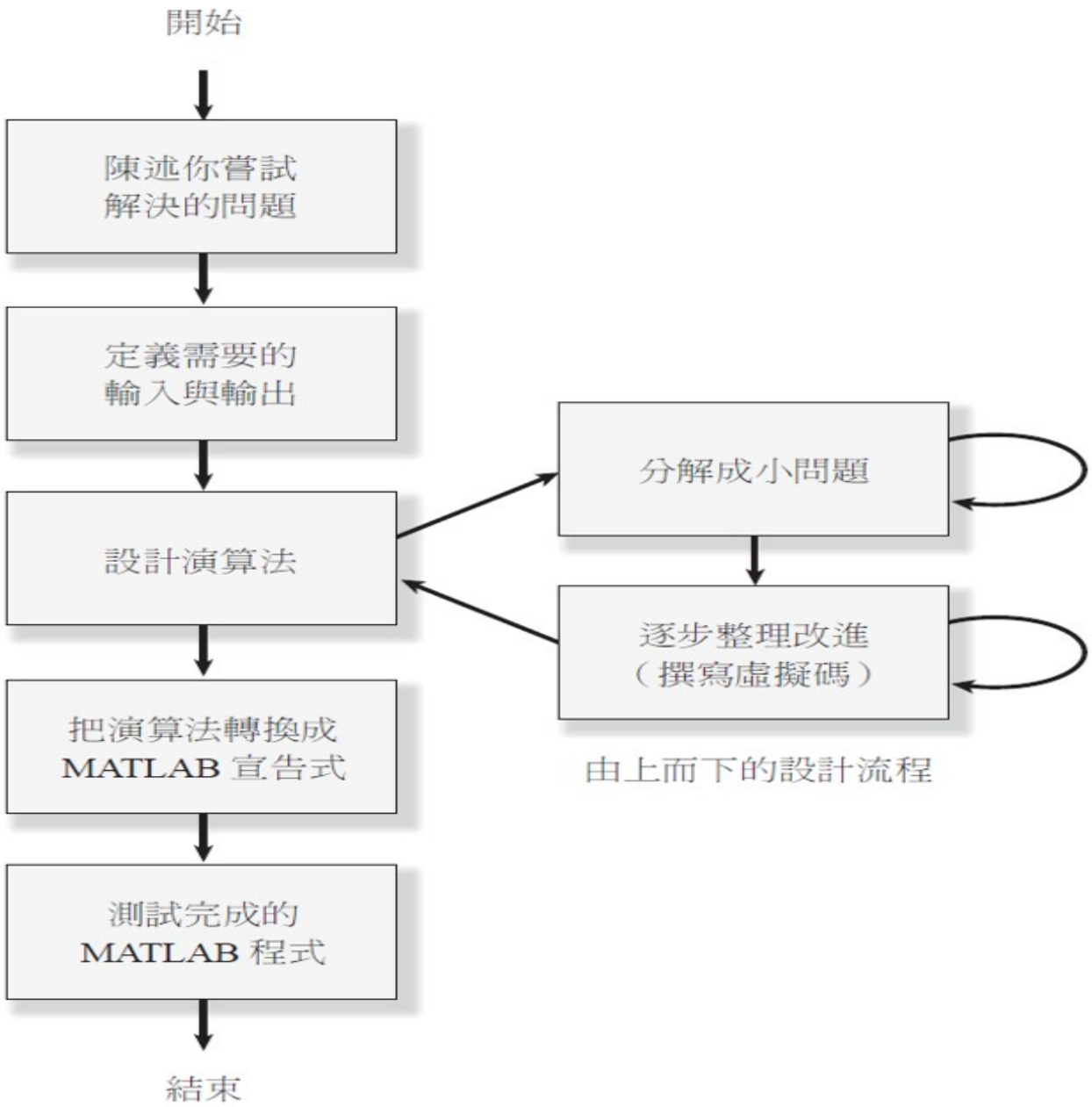
- 4.1 由上而下的設計方法
- 4.2 虛擬碼的使用
- 4.3 邏輯資料型態
- 4.4 分支

Section 4.1
由上而下的設計方法

由上而下的設計 (Top-Down Design)

- 是一種設計流程，首要之務是將一個大型工作計畫分成數個**較小而且較易處理**的子計畫。
- 也是標準化程式設計流程的基礎，這些步驟包括
  1. **清楚地描述想要解決的問題**
  2. **定義程式所需的輸入以及程式所產生的輸出**
  3. **設計想要實施的演算法**
  4. **把演算法轉換成 MATLAB 宣告式**
  5. **測試完成的 MATLAB 程式**

程式設計的流程圖


開始
陳述你嘗試解決的問題
定義需要的輸入與輸出
設計演算法
把演算法轉換成 MATLAB 宣告式
測試完成的 MATLAB 程式
結束
分解成小問題
逐步整理改進（撰寫虛擬碼）
由上而下的設計流程

Section 4.2
虛擬碼的使用

虛擬碼 (Pseudocode)

- 以標準格式描述欲執行的演算法。
- 程式語言 (如 MATLAB、C、FORTRAN) 與英文的混合體。
- 虛擬碼的每一行用簡單又容易理解的英文，來敘述程式設計人員的想法。
- 它的架構如同一般的程式語言，每一行代表一個特定的想法，或是一小段程式碼，但是每一行的敘述都是英文。
- 為什麼要使用虛擬碼?
  - 讓自己及其他人易於了解演算法的設計構想。
  - 將虛擬碼轉換成 MATLAB 程式之前，協助組織自己的想法。
  - 虛擬碼內容十分有彈性，而且容易修改。
  - 虛擬碼對發展演算法是非常有用的!

虛擬碼的範例 (詳見範例 2.3, Sec. 2.12)

INPUT temperature in degrees Fahrenheit. (華氏溫度)

OUTPUT temperature in kelvins. (絕對溫度或是克氏溫度)

Step 1 Prompt user to enter temperature in degrees Fahrenheit.

Step 2 Read temperature in degrees Fahrenheit (`temp_f`).

Step 3 Compute

$$\texttt{temp\_k} \leftarrow \texttt{(5/9) * (temp\_f - 32) + 273.15}.$$

Step 4 Write temperature in kelvins.

MATLAB 程式碼 (承上例)

```
% 提醒使用者輸入華氏溫度
temp_f = input('Enter the temperature (F): ');
% 將華氏溫度轉換為絕對溫度 ( kelvin)
temp_k = (5/9)*(temp_f - 32) + 273.15;
% 輸出計算結果
fprintf('%6.2f (F) = %6.2f (K). \n',temp_f, temp_k);
```

虛擬碼的範例 (一元二次方程式的勘根問題)

To solve the root-finding problem

$$f(x) = ax^2 + bx + c = 0 \qquad \text{with} \quad a \neq 0.$$

INPUT coefficients a, b, c.

OUTPUT approximate root x.

Step 1 Compute the discriminant $D = b^2 - 4ac$.

Step 2 Compute approximate root x to $f(x) = 0$ using D.

Step 3 OUTPUT(x); **STOP**.

虛擬碼的範例 (高斯消去法)

Apply the Gaussian Elimination (GE) to solve a linear system

$$Mx = b,$$

where $M \in \mathbb{R}^{N\times N}$ is invertible and $b \in \mathbb{R}^{N\times 1}$.

INPUT coefficient matrix M and the N-vector b.

OUTPUT approximate solution $x \in \mathbb{R}^{N\times 1}$.

Step 1 Set $A^{(1)} = [M, b] \in \mathbb{R}^{N\times(N+1)}$.

Step 2 Use elementary row operations to compute

$$A = A^{(1)} \to A^{(2)} \to \cdots \to A^{(N-1)} \to A^{(N)},$$

where $A^{(N)}$ is upper triangular.

Step 3 Use the backward substitution to compute x.

Section 4.3
邏輯資料型態

MATLAB 的資料型態

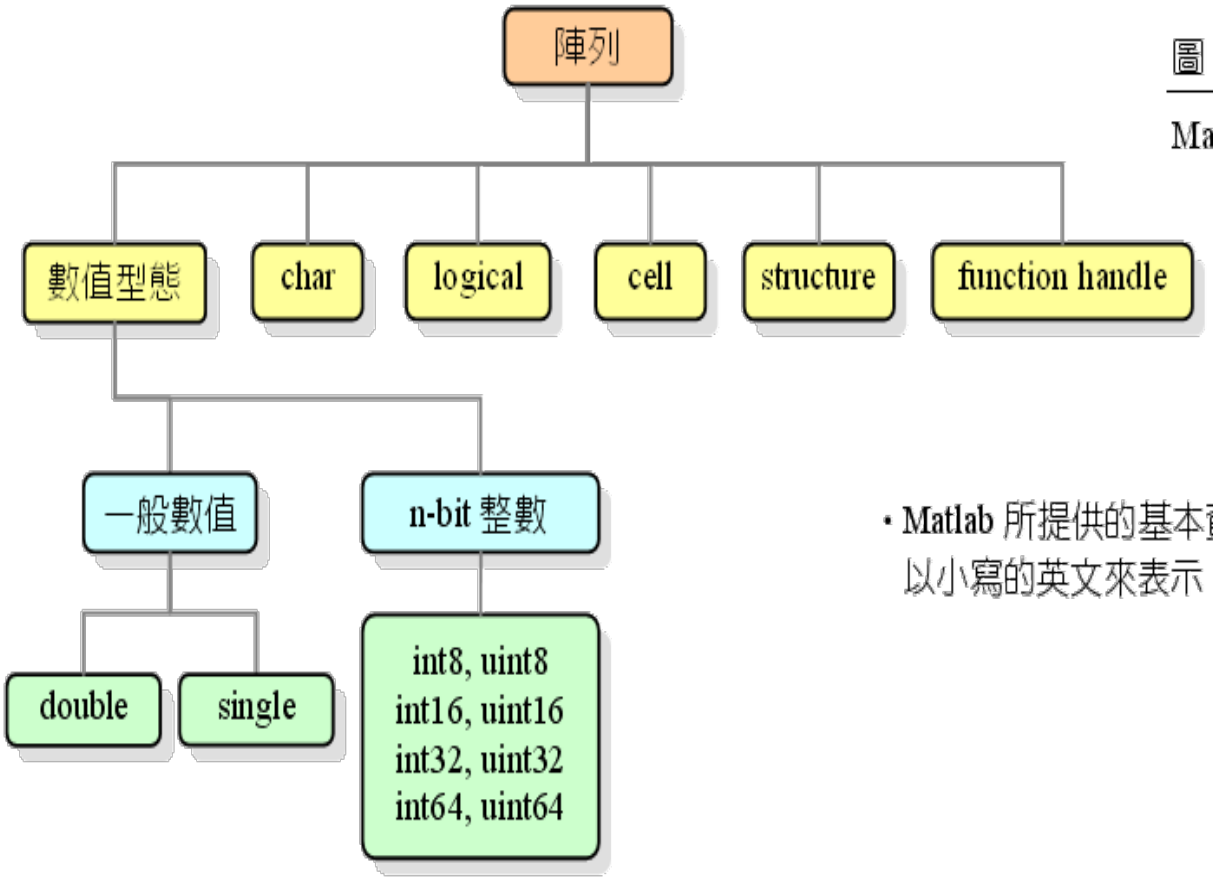


圖 3.1.1

Matlab 提供的資料型態

- Matlab 所提供的基本資料型態以小寫的英文來表示

一般數值 (或浮點數) 資料型態

表 3.1.1　單精度與倍精度型態

資料型態	說 明	位元組	最大的正數	最小的正數
single	單精度	4	3.4028×10^{38}	1.1755×10^{-38}
double	倍精度	8	1.7977×10^{308}	2.2251×10^{-308}

```
>> a = 3.2;   b = single(3.2);
>> whos a b
  Name   Size   Bytes   Class
  a      1×1    8       double
  b      1×1    4       single
```

整數資料型態

n-bit 整數可分為有號（signed）與無號（unsigned）兩種:

表 3.1.2　n-bit 整數型態

資料型態	說明	位元組	最小值	最大值
int8	8-bit 整數	1	−128	127
uint8	8-bit 無號整數	1	0	255
int16	16-bit 整數	2	−32768	32767
uint16	16-bit 無號整數	2	0	65535
int32	32-bit 整數	4	−2147483648	2147483647
uint32	32-bit 無號整數	4	0	4294967295
int64	64-bit 整數	8	−9223372036854775808	9223372036854775807
uint64	64-bit 無號整數	8	0	18446744073709551615

整數資料型態的範例

```
>> uint8([12 300 -250])

ans =

  12  255  0

>> int8(120) + int16(250)
Error using +
Integers can only be combined with integers of the
same class, or scalar doubles.
```

```
>> uint8(12) + uint8(64)

ans =

  76
```

```
>> uint8(12)^3

ans =

  255
```

字元資料型態

- 字元資料型態 (char) 是以**成對的單引號**括起來。
- 每一個字元佔了**兩個位元組** (2 bytes)。

```
>> str = 'a string';
>> asc = double(str)

asc =

  97   32   115   116   114   105   110   103

>> char(asc)

ans =

a string

>> char(65:90)

ans =

ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

邏輯資料型態

- MATLAB 以 **1** 代表運算結果為**真 (true)**，以 **0** 代表運算結果為**假 (false)**。
- 邏輯資料型態的變數佔了一個位元組 (1 byte)。

```
>> a = true
a =
  1  % 不是數字 1 喔!
>> b = 0;  c = logical(b) % 變數 b 是 double 資料型態
c =
  0  % c 為邏輯變數，其內存值是假 (false)
>> logical([5 1 0 -2.4])
ans =
  1  1  0  1 % 非零元素視為 true，零元素視為 false
```

關係運算子 (relational operators)

- 由運算元間的關係產生邏輯（logical）值。
- 形式：　a_1 op a_2

說明	運算子
==	等於
~=	不等於
>	大於
>=	大於或等於
<	小於
<=	小於或等於

範例

```
>> 2 > 4

ans =

  0
```

```
>> 2 <= 4

ans =

  1
```

```
>> a = [1 0;-2 1];
>> b = 0;
>> a > b

ans =

   1   0
   0   1
```

```
>> a = [1 0;-2 1];
>> b = [0 2;-2 -1];
>> a >= b

ans =

   1   0
   1   1
```

關於 == 與 ~= 運算子

- 兩數值以相等運算子 (==) 運算：
  - 相等：傳回 true 值 (1)。
  - 相異：傳回 false 值 (0)。
- 兩數值以不相等運算子 (~=) 運算：
  - 相等：傳回 false 值 (0)。
  - 相異：傳回 true 值 (1)。
- 電腦計算過程會產生**捨入誤差 (roundoff errors)** 導致了相等或不相等測試的失敗。
- 在某個誤差範圍內判斷兩者之間是否幾乎相等。

奇怪的運算結果

% 計算 0.01 連加 10 次的結果。

```
sum = 0;
for ii = 1:10
  sum = sum + 0.01;
end
```

```
>> sum == 0.1

ans =

  0   % 居然是假的 (false)!!
```

在數值計算上，如何判斷兩個浮點數相等?

```
>> a = 0;   b = sin(pi)
b =
   1.2246e-16
```

```
>> a == b
ans =
   0 (false)
```

```
>> abs(a-b) < 1.0e-14
ans =
   1 (true)
```

邏輯運算子

- 作用在一或兩個邏輯運算元的運算子，並產生一個邏輯值。
- 總共有五種二元邏輯運算子：
  - AND (& 與 &&)
  - OR (| 與 ||)
  - 非相容或互斥 OR (xor)
  - 一元運算子：NOT ($\sim$)
- 一般形式：　I_1 op I_2
- 一元邏輯運算形式：　op I_1

邏輯運算子

運算子	說明
&	邏輯 AND
&&	快速求值的邏輯 AND
|	邏輯 OR
||	快速求值的邏輯 OR
xor	邏輯互斥 OR（exclusive OR）
~	邏輯 NOT

邏輯運算子的真值表

輸入敘述		and		or		xor	not
l_1	l_2	l_1 & l_2	l_1 && l_2	l_1 | l_2	l_1 || l_2	xor(l_1, l_2)	~l_1
false	false	false	false	false	false	false	true
false	true	false	false	true	true	true	true
true	false	false	false	true	true	true	false
true	true	true	true	true	true	false	false

重要的 MATLAB 邏輯函式 (1/2)

這些函式可以與關係及邏輯運算子一起用來做為分支及迴圈運算的控制。

函式	目的
false	傳回一個 fase（0）值。
ischar(a)	如果 a 是一個字元陣列，則傳回 true，否則傳回 false。
isempty(a)	如果 a 是一個空陣列，則傳回 true，否則傳回 false。
isinf(a)	如果 a 的值是無限大（Inf），則傳回 true，否則傳回 false。
isnan(a)	如果 a 的值是 NaN（不是數字），則傳回 true，否則傳回 false。
isnumeric(a)	如果 a 的值是數字陣列，則傳回 true，否則傳回 false。
logical	把數字值轉換成邏輯數值：如果數值不是 0，則轉換成 true，如果數值是 0，則轉換成 false。
true	傳回一個 true(1) 值。

重要的 MATLAB 邏輯函式 (2/2)

表 9.1.5　性質測試函數

函 數	說 明
ischar(a)	測試引數 a 是否為一個字元陣列
isempty(a)	測試引數 a 是否為一個空陣列
isequal(a,b)	測試引數 a 與 b 裡的元素個數與數值是否均相等
isfloat(a)	測試引數 a 是否為一個浮點數陣列（包含虛數）
isinteger(a)	測試引數 a 是否為一個 n-bit 整數陣列
islogical(a)	測試引數 a 是否為一個邏輯型態的陣列
isnan(a)	測試引數 a 是否為一個 NaN（not a number）的陣列
isnumeric(a)	測試引數 a 是否為數值（包含 n-bit 整數、實數與虛數）
isprime(a)	測試引數 a 是否為質數
isreal(a)	測試引數 a 是否為實數（含邏輯型態，但不包含 n-bit 整數）
isscalar(a)	測試引數 a 是否為純量（scalars）
issorted(a)	測試引數 a 是否為已經排序好
isspace(a)	測試陣列 a 裡的字元是否為空白字元。若是，則回應 1，否則回應 0
isvector(a)	測試引數 a 是否為一個向量

Section 4.4
分支 (Branches)

分支架構

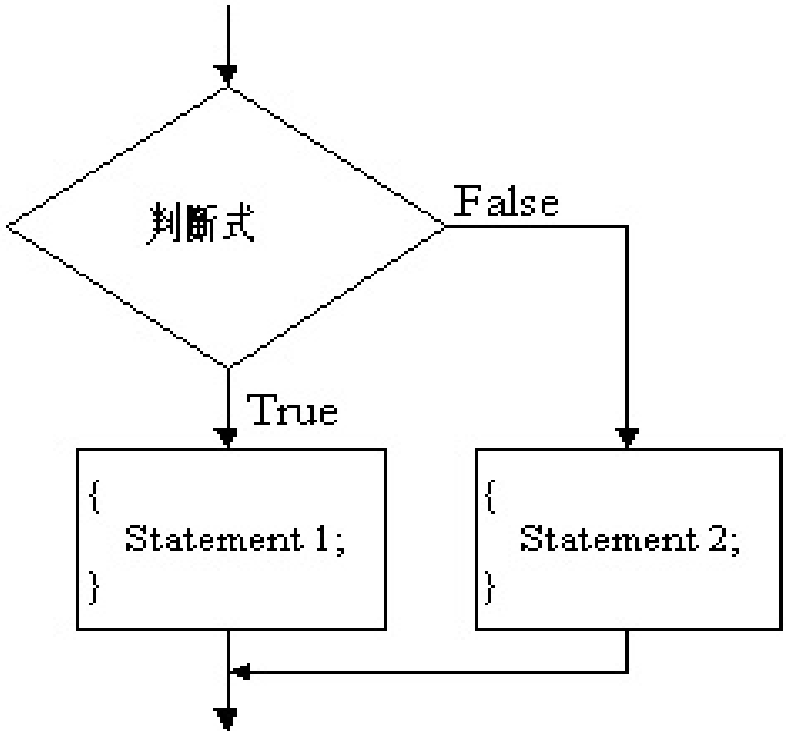
- 分支（branches）是一種 MATLAB 宣告，允許我們選擇想要執行的特定程式區塊（block），而跳過其他部分的程式碼。
- 這些宣告可分成 if 架構、switch 架構，及 try/catch 架構。

if 與 if-else 選擇性敘述

表 9.2.1　if 與 if-else 指令

指令	說 明
if 判斷條件 　　敘述主體 end	若判斷條件為 true，則執行敘述主體
if 判斷條件 　　敘述主體 1 else 　　敘述主體 2 end	若判斷條件為 true，則執行敘述主體 1，否則執行敘述主體 2

if-else 敘述的流程圖



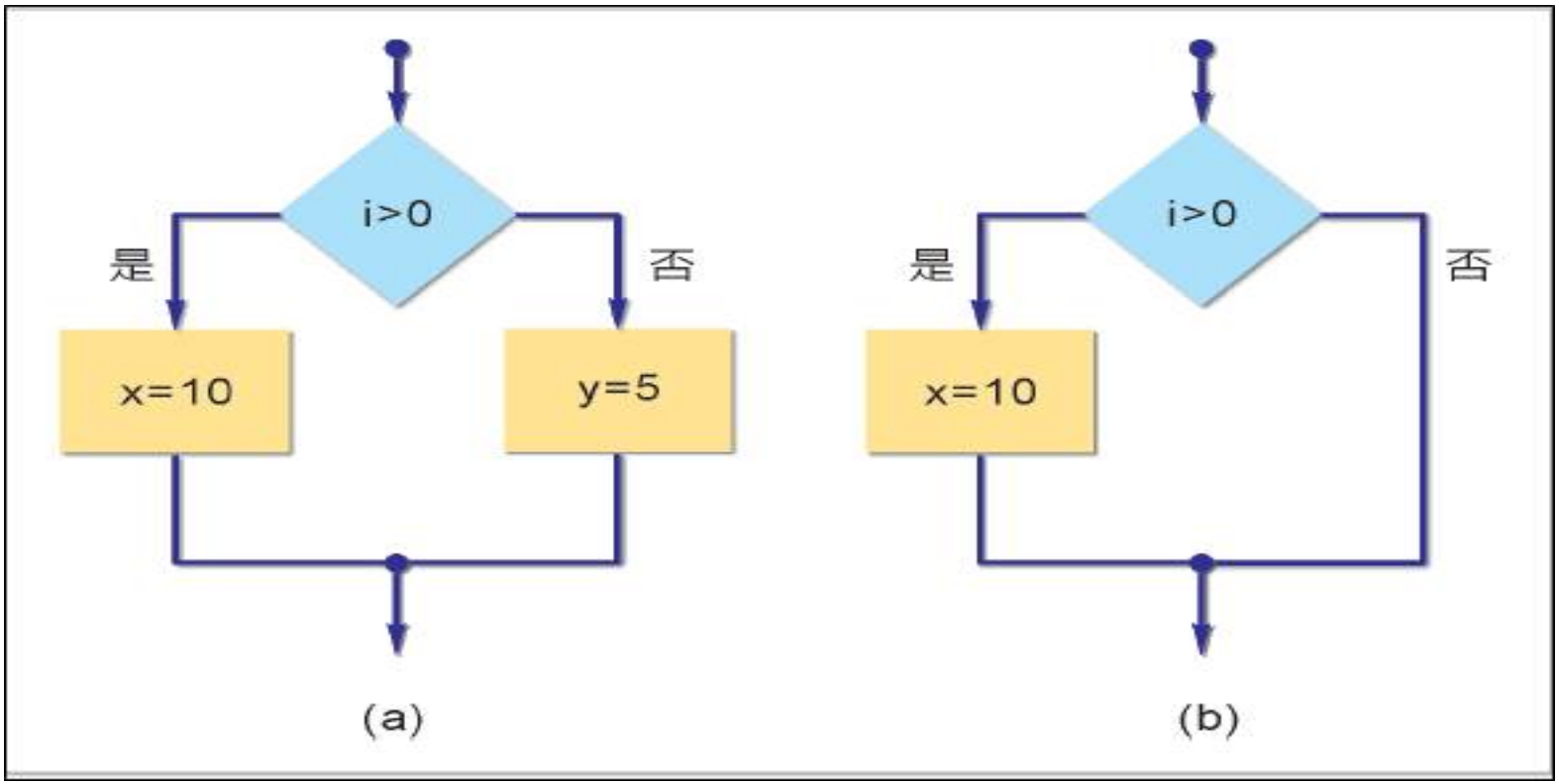
if 敘述的範例

```
i = input(' 請輸入一個數字: '); % i 的內存值被改變!
if i > 0 % 判斷條件
    x = 10; % 敘述主體
end
```

if-else 敘述的範例

```
i = input(' 請輸入一個數字: '); % i 的內存值被改變!
if i > 0 % 判斷條件
    x = 10; % 敘述主體 1
else
    y = 5; % 敘述主體 2
end
```

上述範例的流程圖



if-else 敘述的範例

```
num = input(' 請輸入一個正整數: ');
if mod(num,2) == 0
    fprintf('%g 是偶數 \n',num);
else
    fprintf('%g 是奇數 \n',num);
end
```

if-elseif-else 選擇性敘述

表 9.2.2 **if-elseif-else** 敘述

指 令	說 明

`if` 判斷條件 1
　　敘述主體 1
`elseif` 判斷條件 2
　　敘述主體 2
`elseif` 判斷條件 3
　　敘述主體 3
　　...
`else`
　　敘述主體 n
`end`
	若判斷條件 1 成立，則執行敘述主體 1，否則繼續執行判斷條件 2。若判斷條件 2 成立，則執行敘述主體 2，否則繼續執行判斷條件 3，以此類推。若所有的判斷條件皆不成立，則執行敘述主體 n

if-elseif-else 敘述的範例

```
num = input(' 請輸入變數內容: ');
if isinteger(num)
    disp(' 此變數是 n-bit 整數資料型態');
elseif islogical(num)
    disp(' 此變數是邏輯資料型態');
elseif isfloat(num)
    disp(' 此變數是浮點數資料型態');
else
    disp(' 此變數是其他資料型態');
end
```

範例 4.2: 一元二次方程式 (1/6)

1. 宣告或敘述問題

撰寫 MATLAB 程式，求解一元二次方程式的零根 (zeros or rots)，包含相異的實數根、重根，以及複數根。

2. 定義輸入與輸出

- 鍵入關鍵係數 `a,b,c`。
- 顯示零根解 $\texttt{x}_{1,2} = \dfrac{-\texttt{b} \pm \sqrt{\texttt{b}^2 - 4\texttt{ac}}}{2\texttt{a}}$。

3. 設計演算法

- 讀取輸入資料。
- 計算判別式 $\texttt{discriminant} = \texttt{b}^2 - 4\texttt{ac}$。
- 依據判別式是否大於零、等於零或小於零，各自輸出結果。

範例 4.2: 一元二次方程式 (2/6)

4. 把演算法轉成 MATLAB 宣告式 (檔名: `calc_roots.m`)

```
% Prompt the user for the coefficients
disp('This program solves for the roots of ');
disp('the equation A*X^2 + B*X + C = 0.  ');
a = input('Enter the coefficient A: ');
b = input('Enter the coefficient B: ');
c = input('Enter the coefficient C: ');

% Calculate discriminant
discriminant = b^2 - 4*a*c;
```

範例 4.2: 一元二次方程式 (3/6)

```
if discriminant > 0 % two real roots
    x1 = (-b + sqrt(discriminant)) / (2*a);
    x2 = (-b - sqrt(discriminant)) / (2*a);
    disp('This equation has two real roots:');
    fprintf('x1 = %f \n', x1);
    fprintf('x2 = %f \n', x2);
elseif discriminant == 0 % one repeated root
    x1 = (-b) / (2*a);
    disp('This equation has one repeated root:');
    fprintf('x1 = x2 = %f \n', x1);
else % omplex roots
    re_part = (-b) / (2*a);
    im_part = sqrt(abs(discriminant)) / (2*a);
    disp('This equation has complex roots:');
    fprintf('x1 = %f +i %f \n',re_part,im_part);
    fprintf('x2 = %f -i %f \n',re_part,im_part);
end
```

範例 4.2: 一元二次方程式 (4/6)

5. 測試程式

```
>> calc_roots
This program solves for the roots of
the equation A*X^2 + B*X + C = 0.
Enter the coefficient A: 1
Enter the coefficient B: 5
Enter the coefficient C: 6
This equation has two real roots:
x1 = -2.000000
x2 = -3.000000
```

範例 4.2: 一元二次方程式 (5/6)

```
>> calc_roots
This program solves for the roots of
the equation A*X^2 + B*X + C = 0.
Enter the coefficient A: 1
Enter the coefficient B: 4
Enter the coefficient C: 4
This equation has one repeated root:
x1 = x2 = -2.000000
```

範例 4.2: 一元二次方程式 (6/6)

```
>> calc_roots
This program solves for the roots of
the equation A*X^2 + B*X + C = 0.
Enter the coefficient A: 1
Enter the coefficient B: 2
Enter the coefficient C: 5
This equation has complex roots:
x1 = -1.000000 +i 2.000000
x2 = -1.000000 -i 2.000000
```

switch-case-otherwise 敘述

表 9.2.3　switch-case-otherwise 敘述

指 令	說 明
switch 運算式 　case 選擇值 1 　　敘述主體 1 　case 選擇值 2 　　敘述主體 2 　　... 　otherwise 　　敘述主體 n end	若運算式的值等於選擇值 1，則執行敘述主體 1，若運算式的值等於選擇值 2，則執行敘述主體 2，以此類推。如果運算式的值皆不等於所列的選擇值，則執行敘述主體 n 若接在 case 後面的選擇值不只一個時，可用大括號將它們括起來，如 {選擇值 1, 選擇值 2, ..., 選擇值 n}

Remarks

- switch 架構是另一種分支架構的形式。
- switch 架構中的運算式或選擇值允許依據
  1. 整數值
  2. 字元字串
  3. 邏輯敘述

  選擇執行某個特定程式區塊。

switch-case-otherwise 敘述的範例

```
value = input(' 請輸入介於 1 到 10 之間的正整數: ');
switch value
  case {1,3,5,7,9}
    disp(' 此數字是一個奇數。');
  case {2,4,6,8,10}
    disp(' 此數字是一個偶數。');
  otherwise
    disp(' 此數字不在預設範圍內喔!');
end
```

switch-case-otherwise 敘述的範例

```
method = input('Enter the name of method:  ','s');
switch method
  case {'linear','bilinear'}
    disp('Linear/Bilinear method.');
  case 'cubic'
    disp('Cubic method.');
  otherwise
    disp('Unknown method!');
end
```

Thank you for your attention!