16 程式編寫概念

目標

在完成這章後，你將能夠

- 了解變數的應用及如何改變它們的數值
- 追蹤使用偽代碼程式的邏輯流程
- 追蹤變數程式流程圖或程式片段執行期間的數值轉變
- 認識電腦程式的基本結構，包括分支語句、條件語句和迭代語句

在這章中，我們將學習基本的程式編寫概念，包括**算術運算**和程式的**邏輯流程**。為免處理真正的程式語言，我們將通過使用稱為**偽代碼 (Pseudocode)** 的簡短語句來學習程式編寫的概念。

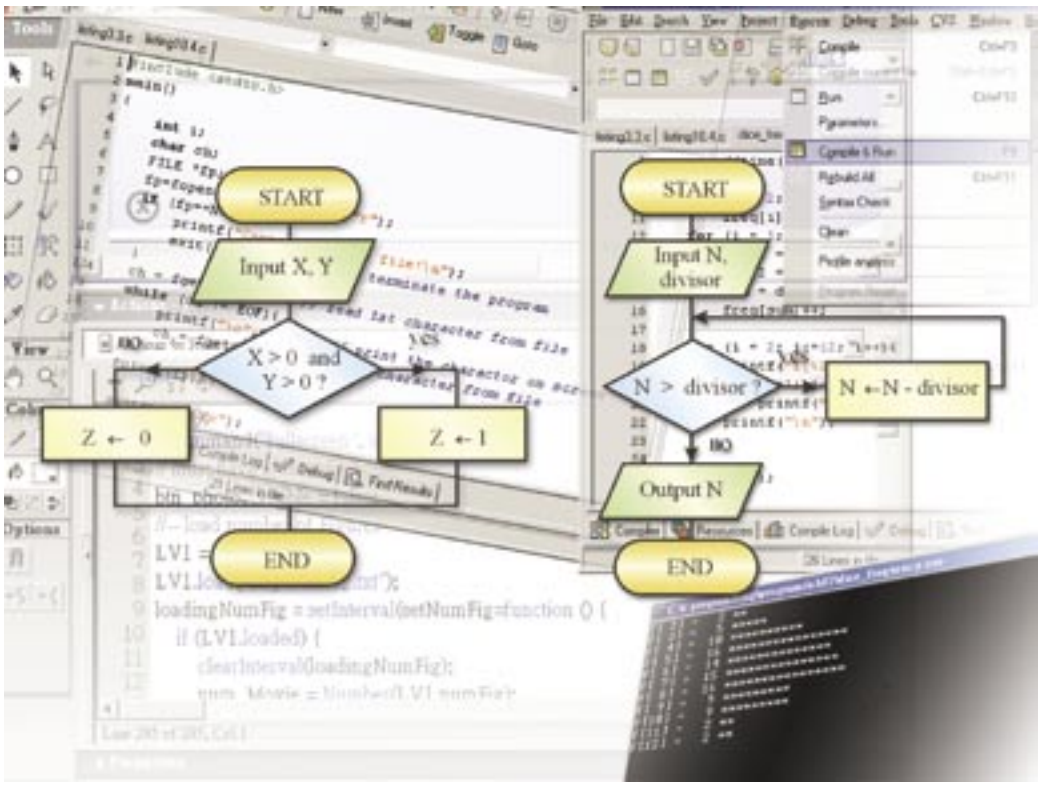
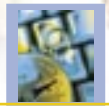


圖 1　程式編寫概念

分支語句 branching statement　條件語句 conditional statement　迭代語句 iterational statement
算術運算 arithmetic operation　邏輯流程 logic flow



16.1 變數和算式

重　點

變數代表一個保存數據的存貯格。

賦值語句把數值存貯在變數中，並會重寫它的內容。

A. 變數

在程式由電腦執行期間，記憶體上某些存貯格的內容會不斷地更改。這些存貯格是以變數名字來識別。因此，**變數 (Variable)** 可視為一個記憶體存貯格，用於存貯數字、常數、字串等數據。

重　點

一般數據的類型包括：

1. 數字
2. 字元

賦值語句 (Assignment statement) 把數值存貯於變數中，並重寫變數先前的內容。在本書中，我們將使用箭咀符號 "←" 把右邊的數值或運算結果存貯在左邊的變數中。其他語言對應的符號是 PASCAL 的 ":=" 及 C 和 BASIC 的 "="。

賦值語句的格式是

```
變數 ← {數值或算式}
```

表 1 提供一些賦值語句例子：

	賦值語句	描述
1.	`X ← 5`	把整數 5 放在變數 X 內。
2.	`Y ← 2.5`	把實數 2.5 放在變數 Y 內。
3.	`name ← "Peter"`	把字串 "Peter" 放在變數 name 內。

表 1　賦值語句的例子

一般的數據包括兩類型：**數字 (Numeric)** 和**字元 (Character)**。數字數據用於計算，包括整數和實數，例子是 2, -3, 5.2, 0.0006；字元數據用於顯示信息等內容，並置於引號內，例如 "A", "2.3", "Peter" 等。任何從鍵盤鍵入的都可以是字元數據。

存貯格　memory location
整數　integer
數字數據　numeric data
實數　real number
字元數據　character data

B. 算術運算

重點

電腦首先計算「乘」、「除」，然後才計算「加」、「減」。

算術運算 (Arithmetic operation) 包括加、減、乘、除，所用的**運算符 (Operator)** 分別是＋、－、＊、/。表 2 提供一些算術運算例子：

	語句	結果
1.	X ← 5 + 8	X 存貯 13
2.	X ← 5 - 7	X 存貯 -2
3.	X ← 3 * 2	X 存貯 6
4.	X ← 8 / 2	X 存貯 4
5.	X ← 5 / 2	X 存貯 2.5

表 2　算術運算的例子

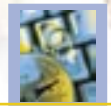
箭咀右邊的運算組合稱為**算式 (Expression)**，運算的結果會放入左邊的變數中。

C. 運算符優次

算式在電腦的運算次序與數學上的**運算符優次 (Order of precedence)** 相同，即首先計算乘除，然後才計算加減，所謂「先乘除後加減」。細閱下列的例子：

	語句	結果
1.	X ← 4 * 2 + 3	由於 4*2 等於 8，語句變成 X ← 8 + 3. 因此，X 存貯 11。
2.	X ← 4 + 2 * 3	先進行「乘」，才做「加」。 由於 2*3 和 6 相等，語句變成 X ← 4 + 6. 因此，X 存貯 10。
3.	X ← 4 * 2 / 8	由於 4*2 等於 8，語句變成 X ← 8 / 8. 因此， X 存貯 1。
4.	X ← 4 + 8 / 4	先進行「除」，然後才做「加」。 由於 8 / 4 等於 2，語句變成 X ← 4 + 2. 因此， X 存貯 6。
5.	X ← 3 * 7 - 2 / 5	先進行「乘」和「除」，然後才做「減」。 語句變成 X ← 21 - 0.4. 因此，X 存貯 20.6。

表 3　運算符優次的例子



若算式包含括弧，則電腦會優先計算括弧內的**子算式**。詳細閱讀下列的例子：

	語句	結果
1.	X ← 7 - (2 + 3)	語句變成 X ← 7 - 5. 因此，X 存貯 2。
2.	X ← (-2 * 5) + 6	語句變成 X ← -10 + 6. 因此，X 存貯 -4。
3.	X ← 3 * (-2 - 3)	語句變成 X ← 3 * (-5). 因此，X 存貯 -15。

表 4　括弧內的算式會被優先計算

D. 較複雜的算式

算式可能包含多個變數。假設在程式執行前，X 存貯 5、Y 存貯 -3：

	語句	結果
1.	X ← X + 2	語句變成 X ← 5 + 2. 因此，X 存貯 7。.
2.	X ← Y * 2	語句變成 X ← (-3) * 2. 因此，X 存貯 -6。
3.	Y ← X * Y + Y	語句變成 Y ← 5 * (-3) + (-2). 因此，Y 存貯 -17。
4.	X ← Y * 2 Y ← X + Y	第一語句變成 X ← (-3) * 2，也就是 X 存貯 -6，第二個語句變成 Y ← -6 + (-3). 因此，Y 存貯 -9 而 X 存貯 -6。
5.	X ← X + Y Y ← X + Y	第一語句變成 X ← 5 + (-3) ，也就是 X 存貯 2。第二語句變成 Y ← 2 + (-3). 因此， Y 存貯 -1 而 X 存貯 2。

表 5　多於一個變數的賦值語句

括弧 parentheses　　子算式 subexpression

E. 字串接法

字串接法 (String concatenation) 意謂把兩個**字串 (String)** 或**字符串**，結合在起來成為單一字串，使用的**運算符**是 "+"。

	語句	結果
1.	A ← "2" + "3"	A 存貯 "23".
2.	A ← "Hi" + "!"	A 存貯 "Hi!".
3.	A ← "Pet" + "er Pan"	A 存貯 "Peter Pan".

表 6　字串接法的例子

以下的表格列出字串接法的一些例子，假設最初變數 A 存貯 "3"、B 存貯 "27"、C 存貯 "Peter"。

	語句	結果
1.	X ← A + "2"	語句變成 X ← "3" + "2". 因此，X 存貯 "32". 注意: A 的內容不變。
2.	X ← "2" + A	語句變成 X ← "2" + "3". 因此，X 存貯 "23".
3.	X ← A + A + A	語句變成 X ← "3" + "3" + "3". 因此，X 存貯 "333".
4.	X ← A + B	語句變成 X ← "3" + "27". 因此，X 存貯 "327".
5.	C ← C + B	語句變成 C ← "Peter" + "27". 因此，C 存貯 "Peter27".

表 7　更多的字串接法例子

重　點

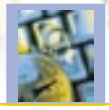
字串和數字之間的運算是無意義的。

以下的語句是無效的：

```
X ← "A" + 2
```

原因是字串和數字之間的運算是無意義的。

運算符 operator　　無效 invalid



16.2　輸入和輸出語句

重　點

輸入語句讓用戶互動式更改變數的值。

輸出語句把數據顯示在屏幕上。

A.　輸入語句

輸入語句 (Input statement) 接受用戶輸入的數據，並將數據存貯在變數中。輸入語句的格式是

```
INPUT {變數}
```

例如　　INPUT X

若用戶在鍵盤上對以上輸入語句鍵入 2.5 時，X 將存貯 2.5。

B.　輸出語句

輸出語句 (Output statement) 把數據顯示在屏幕上。輸出語句的格式是

```
OUTPUT {算式}
```

例如，下列的輸出語句會把 2.5 和 2 的積，即 5，顯示在屏幕上：

```
OUTPUT 2.5*2
```

在下列各項例子中，我們將假設 X 存貯 3：

	語句	結果
1.	OUTPUT X	輸出是 3
2.	OUTPUT 2 * 4 + 5	輸出是 13.
3.	OUTPUT 2 * X + 4	電腦將計算 2 * 3 + 4. 輸出是 10.

表 8　　輸出語句的例子

積 product

例 1　　下列的程式片段將會計算三角形的面積。

```
10  INPUT HEIGHT
20  INPUT BASE
30  AREA ← HEIGHT * BASE / 2
40  OUTPUT AREA
```

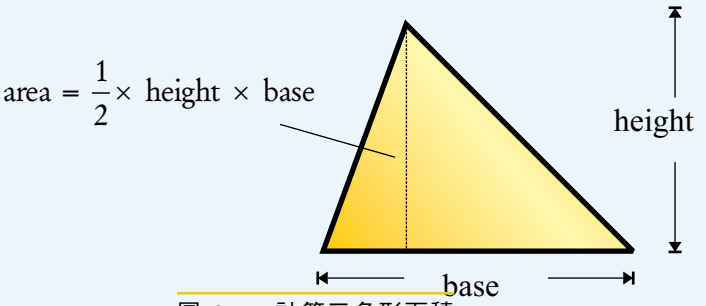


圖 2　計算三角形面積

若用戶以輸入 5 和 4 分別回應第 10 和 20 行，輸出將會是 10。

例 2　　下列的程式片段將會把兩個輸入的數據互相交換。

```
10  INPUT X
20  INPUT Y
30  Z ← X
40  X ← Y
50  Y ← Z
60  OUTPUT X
70  OUTPUT Y
```

交換 X 和 Y

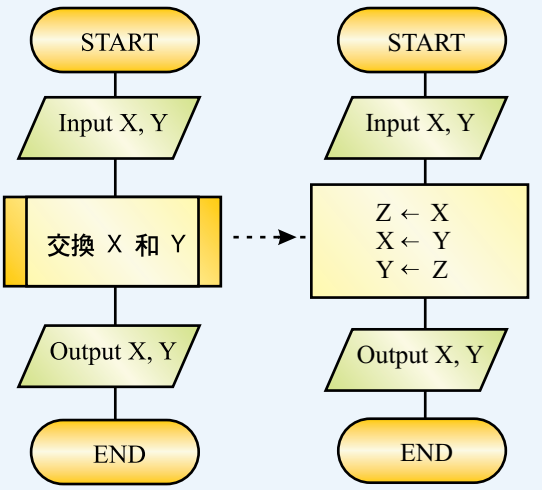
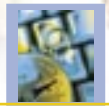


圖 3　兩個變數的交換

第 30 到 50 行的語句組目的是將 X 和 Y 的內容互相交換（或稱**調換 Swap**）。假設用戶輸入了 3 和 6，在第 30 行後，X 的數值會複製到 Z 中，因此，Z 會存貯 3；在第 40 行後，Y 的數值會複製到 X，因此，X 會存貯 6；在第 50 行之後，Z 的數值會複製到 Y，因此，Y 會存貯 3，見圖 4 的說明。結果，上述的程式輸出將是 6 和 3。

原本數值	X 3　Y 6　Z
30　Z ← X	X 3　Y 6　Z 3
40　X ← Y	X 6　Y 6　Z 3
50　Y ← Z	X 6　Y 3　Z 3

圖 4　變數 X 和 Y 的交換



16.3　條件語句

重　點

條件語句可能是

IF ... THEN ... ELSE ... ENDIF

或

IF ... THEN ... ENDIF

條件語句 (Conditional statement) 是指在符合指定條件的情況下，電腦才會執行某些行動。條件語句提供一個或以上的**替代選擇 (Alternatives)** 讓程式決定。圖 5 的流程圖稱為**選擇控制結構 (Selection control structure)**。

條件語句的格式，其中之一是

```
IF {條件算式} THEN
    {行動 1}
ELSE
    {行動 2}
END IF
```

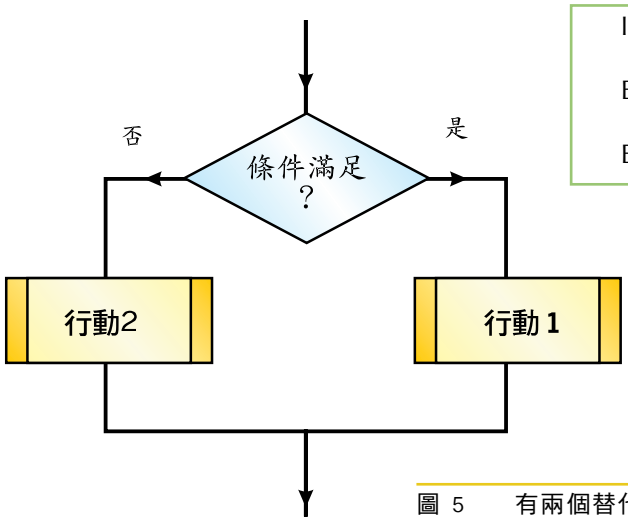


圖 5　有兩個替代選擇的選擇控制結構

經運算後，**條件算式 (Conditional expression)** 的結果只有兩個可能：「真」或「假」。若結果是「真」，「行動 1」才會執行，否則「行動 2」便會執行。每個行動可超過一句語句。

例 3　下列的程式片段將會根據輸入的數值，將不同的數值分配到變數 Y。

```
10    INPUT X
20    IF X > 0 THEN
30        Y ← 6
40    ELSE
50        Y ← -10
60    END IF
```

若用戶在回應第 10 行的輸入語句時，鍵入一個正數，例如 5 或 12，Y 的數值將會是 6；若用戶鍵入 0 或負數，例如 -5 或 -12，Y 的數值將會是 -10。

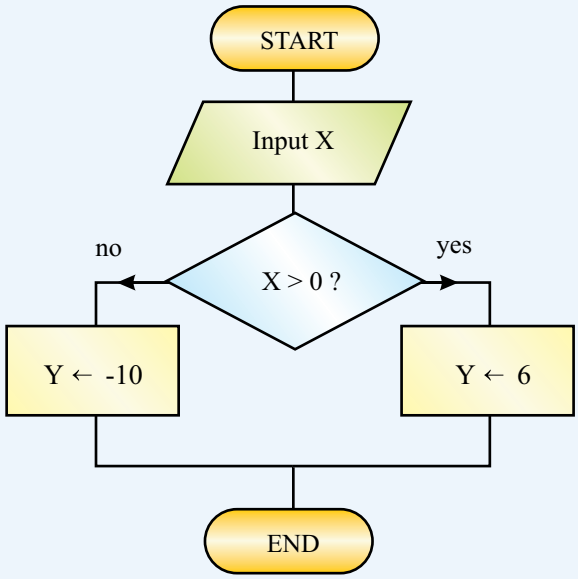


圖 6　根據 X 的正負值作決定

「真」 true　　「假」 false

A. 關係運算符

在上述的例子中，條件算式內的符號 ">" 稱為**關係運算符 (Relational operator)**，意謂「大於」，其他的關係運算符包括 "="、">="、"<="、"<" 及 "<>"。

表 9　關係運算符

關係運算符	意義
=	等於
<>	不等於
>	大於
<	小於
>=	大於或等於
<=	小於或等於

例 4　下列的程式片段會得出與例 3 相同的效果。

```
10  INPUT X
20  IF X <= 0 THEN
30      Y ← -10
40  ELSE
50      Y ← 6
60  END IF
```

若輸入的數值小於或等於零，Y 的數值將會是 -10。若輸入的數值大於零，Y 的數值將會是 6。

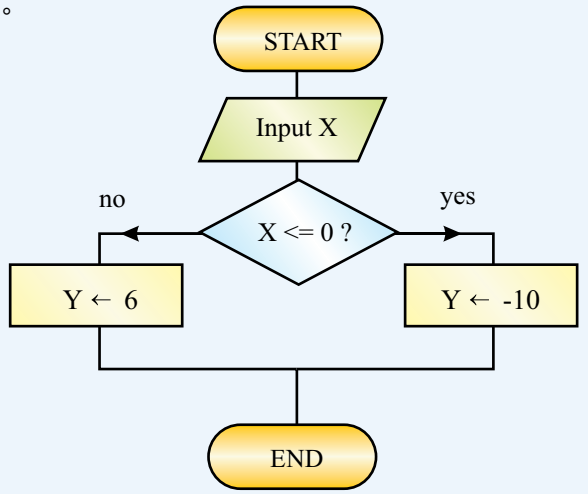


圖 7　根據 X 的正負值作決定

例 5　下列的程式片段將會把最大的輸入數值放入變數 Z 中。

```
10  INPUT X
20  INPUT Y
30  IF X > Y THEN
40      Z ← X
50  ELSE
60      Z ← Y
70  END IF
```

若用戶輸入 5 和 8，由於第 30 行的條件不能滿足，所以第 60 行的行動將會執行，也就是 Z ← Y。因此，Z 將會存貯 8，即較大的輸入數值。

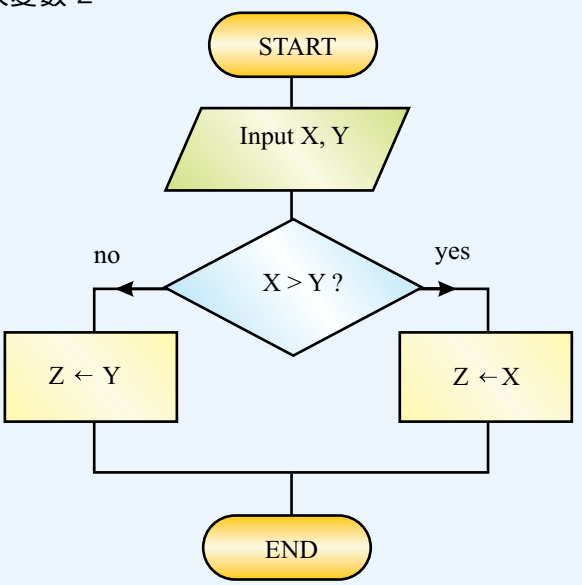


圖 8　以 X 和 Y 的相關數值為基礎的選擇

B. 只有一個替代選擇的條件語句

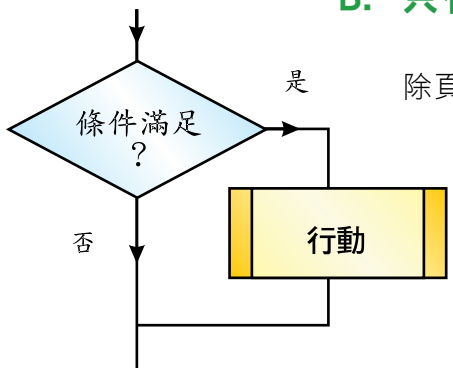


圖 9　只有一個替代選擇的選擇控制結構

除頁 103 的格式外，條件語句的另一個格式是

```
IF {條件算式} THEN
    {行動}
END IF
```

只有當條件算式的結果是「真」時，「行動」才會執行，否則不會執行條件語句內的任何行動。

例 6　下列的程式片段將顯示輸入數據的絕對值。

```
10  INPUT X
20  IF X < 0 THEN
30      X ← -X
40  END IF
50  OUTPUT X
```

第 30 行的符號 "-" 稱為**「單元運算符」(Unary operator)**，用於改變一個變數的符號，故第 30 行等同下列的語句：

```
30  X ← -1 * X
```

若輸入是負數，例如 -5，將會乘以 -1，並得出 5。這個程式的結果總是正數或零，故稱為**絕對值**。

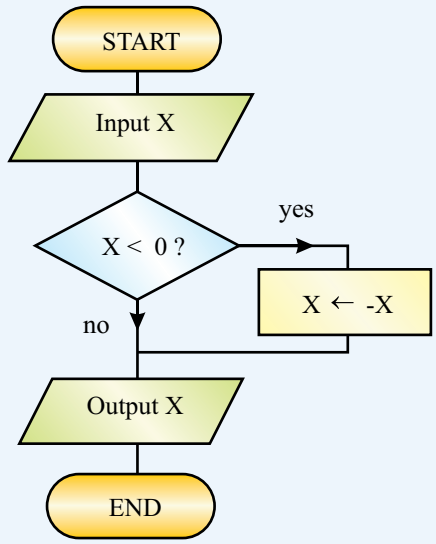


圖 10　計算絕對值

例 7　下列的程式片段將輸入兩個數據，並為它們由小至大排序。

```
10  INPUT X
20  INPUT Y
30  IF X > Y THEN
40      Z ← X
50      X ← Y
60      Y ← Z
70  END IF
80  OUTPUT X
90  OUTPUT Y
```

在第 30 行中，電腦比較 X 和 Y 的內容，若 X 大於 Y，第 40 到 60 行的行動將會執行，也就是將 X 和 Y 的數值調換，結果是 X 總是小於 Y。

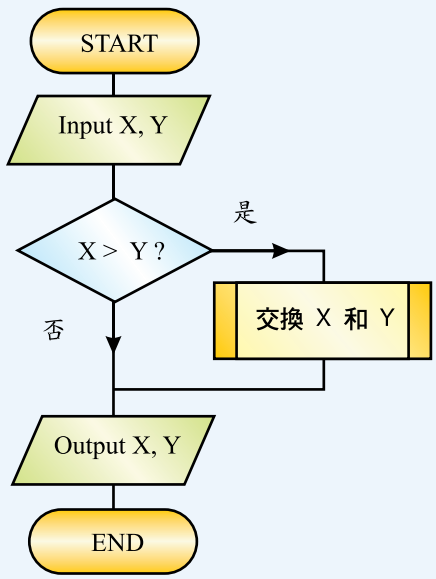


圖 11　為兩個數據由小至大排序

行動 action　　絕對值 absolute value

C. 布爾運算符

如上文所述，條件算式的結果只有「真」或「假」，這類結果稱為**布爾值 (Boolean value)**。

條件算式可涉及多於一個條件。舉例來說，要購買某件貨品，你必須喜歡該貨品**及**你有足夠的金錢，以上兩個條件必須同時滿足。上述例子中的「及」連繫兩個布爾值，並提供新的布爾值。在程式編寫時，「及」是以**布爾運算符 (Boolean operator)** "AND" 來代表，其他的**布爾運算符**包括「或」"OR" 和「非」"NOT"。

例 8　下列的程式片段在兩個輸入數據都是正數時，才會把 1 賦值到 Z，否則，將 0 賦值到 Z。

```
10  INPUT X
20  INPUT Y
30  IF X > 0 AND Y > 0 THEN
40      Z ← 1
50  ELSE
60      Z ← 0
70  END IF
```

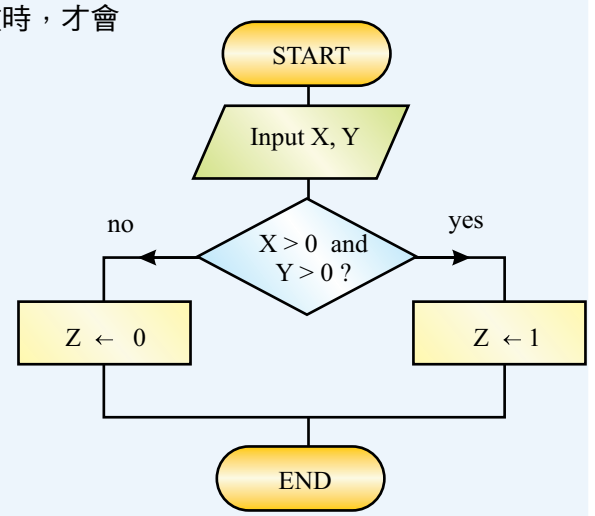


圖 12　測試兩個數字均是正數

布爾運算符 "AND" 使用下列**真值表 (Truth table)** 的規則來求得結果。假設 p 和 q 分別代表兩個布爾值：

	p	q	p AND q
1.	假	假	假
2.	假	真	假
3.	真	假	假
4.	真	真	真

表 10　布爾運算符 "AND" 的真值表

在例 8 中，假設輸入的數據是 5 和 -4，電腦首先決定每個關係運算的結果，然後使用表 10 中的第 3 行來決定最後結果，由於最後結果是「假」，電腦把 0 賦值到 Z。

布爾值 Boolean value　　布爾運算符 Boolean operator

	p	q	p OR q
1.	假	假	假
2.	假	真	真
3.	真	假	真
4.	真	真	真

表 11　布爾運算符 "OR" 的真值表

例 9　在下列的程式片段中，若兩個輸入的數值其中一個是正數，Z 將獲賦值 1，否則，Z 將獲賦值 0。

```
10  INPUT X
20  INPUT Y
30  IF X > 0 OR Y > 0 THEN
40      Z ← 1
50  ELSE
60      Z ← 0
70  END IF
```

若用戶鍵入 5 和 -4，第一個條件結果是「真」，而第二個條件是「假」。電腦將使用表 10 的第 3 行來決定最後結果，由於最後的結果是「真」，電腦便把 1 賦值到 Z。

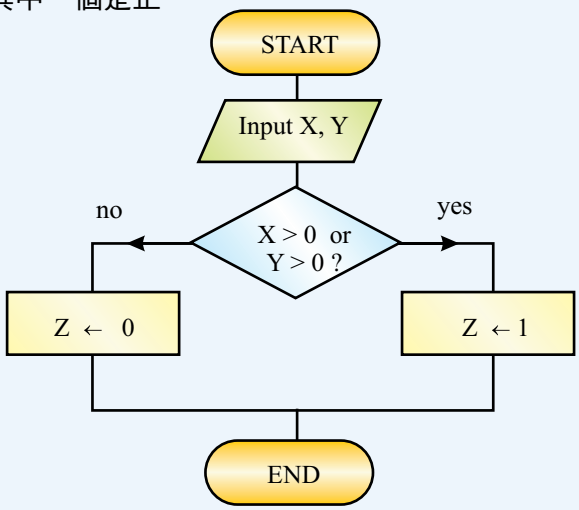


圖 13　測試輸入的數據是否其中一個是正數

	p	NOT p
1.	假	真
2.	真	假

表 12　布爾運算符 "NOT" 的真值表

例 10　在下列的程式片段中，若兩個輸入的數值不相同，Z 將獲賦值 1，否則，將獲賦值 0。

```
10  INPUT X
20  INPUT Y
30  IF NOT (X = Y) THEN
40      Z ← 1
50  ELSE
60      Z ← 0
70  END IF
```

第 30 行等同下列的語句：

```
30  IF X <> Y THEN
```

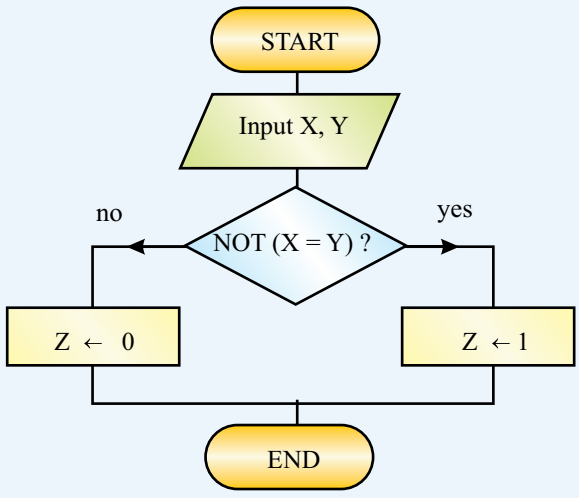


圖 14　測試輸入的數據是否相同

16.4 分支語句

重　點

分支語句可以改變程式執行的次序。

分支語句 (Branching statement) 可改變程式的執行次序，並以 GOTO 指令來實踐。分支語句的格式是

```
GOTO {行數}
```

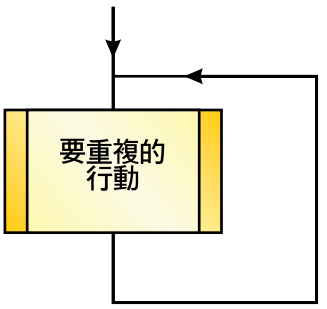


圖 15　分支法（無限循環）

下列是一個分支語句的例子：

```
10  X ← 0
20  X ← X + 1
30  GOTO 20
```

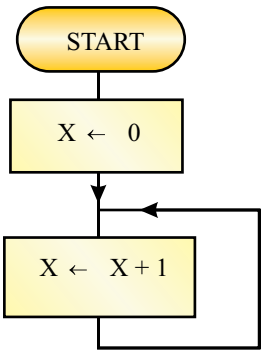


圖 16　無限循環的例子

上述程式將不會自動終止，電腦將不斷地把 X 的數值增加 1，令 X 從 0, 1, 2, ...不斷地改變。我們稱程式掉進了**無限循環 (Infinite loop)**，電腦將會失去反應直至人手終止程式。

分支語句應該與條件語句一起使用，像下列的例子般：

```
10  INPUT X
20  IF X < 0 THEN
30      GOTO 10
40  END IF
50  OUTPUT X
```

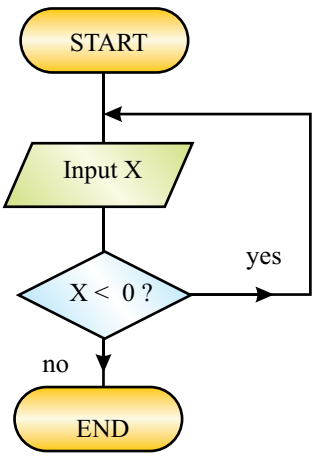
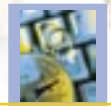


圖 17　條件分支（迭代）

在上述程式中，電腦若發現輸入的數據是負數 (X < 0)，它將重複要求用戶輸入，以確保數據值不小於零（即大於或等於零）。

人手終止　manually stopped



例 11　　下列的程式片段將會把 X 的數值多次重複地增加 1。

```
10    X ← 0
20    X ← X + 1
30    IF X < 3 THEN
40        GOTO 20
50    END IF
60    OUTPUT X
```

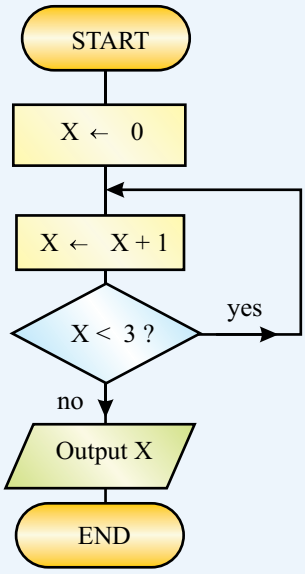


圖 18　進行三次的迭代

當第 20 行在第一次執行後，X 將會存貯 1，由於第 30 行的條件得到滿足，電腦將分支到第 20 行，並再次執行。以上動作將會不斷重複直到 X 等於 3。

在執行第 30 行後	X 的結果	第 30 行的條件	行動
第一次	1	滿足 (1 < 3)	GOTO 20
第二次	2	滿足 (2 < 3)	GOTO 20
第三次	3	不滿足 (3 = 3)	OUTPUT X

因此，程式的輸出是 3。

注意，第 20 行總共執行了三次，每次稱為一個**迭代 (Iteration)** 或稱**重複**，因此，上述程式有三個迭代。我們將會在 16.5, 16.6, 16.7 中學習一些有指定執行次數的迭代結構。

例 12　　細閱下列的程式片段：

```
10  S ← 0
20  X ← 0
30  X ← X + 1
40  S ← S + X
50  IF X < 10 THEN
60      GOTO 30
70  END IF
80  OUTPUT S
```

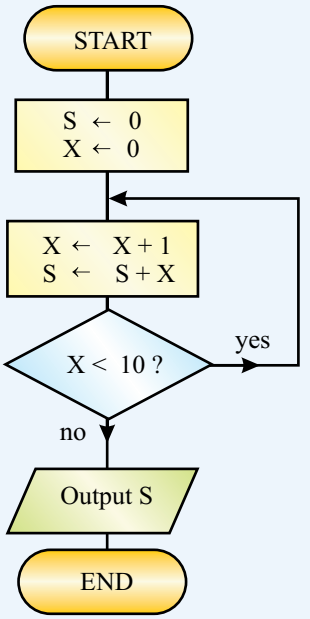
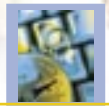


圖 19　進行＋次的迭代

在上述程式中，只要 X 小於 10，第 30 和 40 行將不斷重複。第 30 行將 X 增加 1，而第 40 行計算 X 的總和，即是將每次迭代的 X 加起來並存貯在 S 中。由於 X 從 1 改變到 10，這個程式目的是計算 1 + 2 + … + 10 的總和，因此預計輸出是 55。

在執行第 50 行後	X 的結果	S 的結果	第 30 行的條件	行動
第一次	1	1	滿足 (1 < 10)	GOTO 30
第二次	2	3	滿足 (2 < 10)	GOTO 30
第三次	3	6	滿足 (3 < 10)	GOTO 30
第四次	4	10	滿足 (4 < 10)	GOTO 30
第五次	5	15	滿足 (5 < 10)	GOTO 30
第六次	6	21	滿足 (6 < 10)	GOTO 30
第七次	7	28	滿足 (7 < 10)	GOTO 30
第八次	8	36	滿足 (8 < 10)	GOTO 30
第九次	9	45	滿足 (9 < 10)	GOTO 30
第十次	10	55	不滿足 (10 = 10)	OUTPUT S



16.5　For 迴路

重　點

For 迴路以指定的次數執行迭代。

例 11 和 12 示範了兩個有指定重複次數的迭代。事實上，它們可以由較簡單的「**For 迴路**」取代。For 迴路的格式是：

```
FOR {變數} = {開始值} TO {結束值} STEP {步伐}
    {要重複的動作}
NEXT
```

若保留字 STEP (步伐) 給省略掉，程式會使用預設為 1 的步伐。

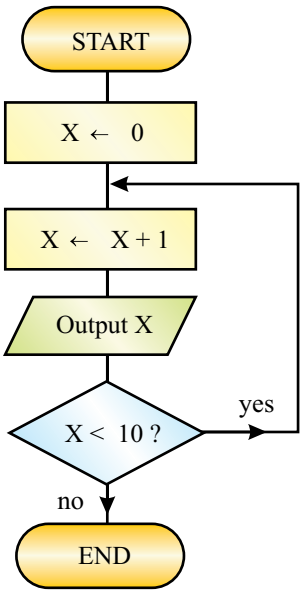


圖 20　進行＋次迭代的 For 迴路

```
FOR {變數} = {開始值} TO {結束值}
    {要重複的動作}
NEXT
```

細閱下列例子：

```
10    FOR X = 1 to 10
20        OUTPUT X
30    NEXT
```

由於 X 從 1 增加到 10，程式將會輸出 1, 2, 3, ... 10。

例 13　下列的程式將會計算

$1^2 + 2^2 + 3^2 .... + 10^2$ 的總和。

```
10    S ← 0
20    FOR X = 1 TO 10
30        S ← S + X * X
40    NEXT
50    OUTPUT S
```

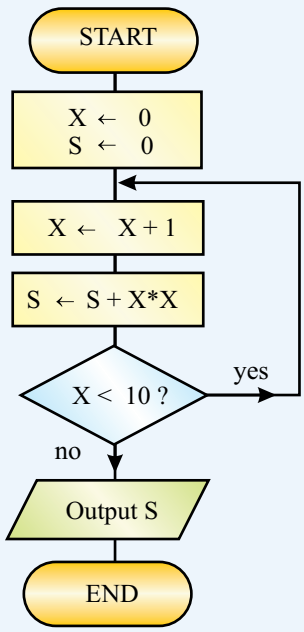


圖 21　計算整數從 1 到 10 平方的總和

迴路 loop

例 14　下列的程式將會計算

$$2^1 + 2^2 + 2^3 .... + 2^{10}$$

的總和。

```
10  S ← 0
20  P ← 1
30  FOR X = 1 TO 10
40      P ← P * 2
50      S ← S + P
60  NEXT
70  OUTPUT S
```

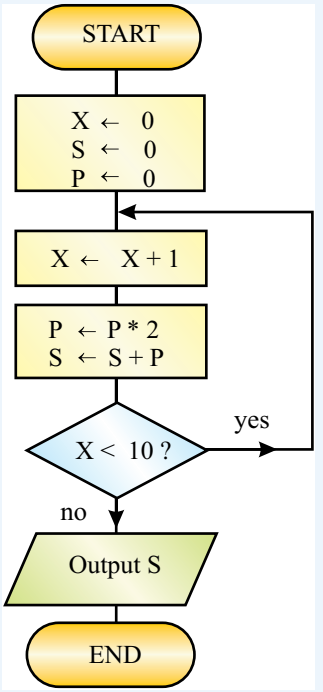


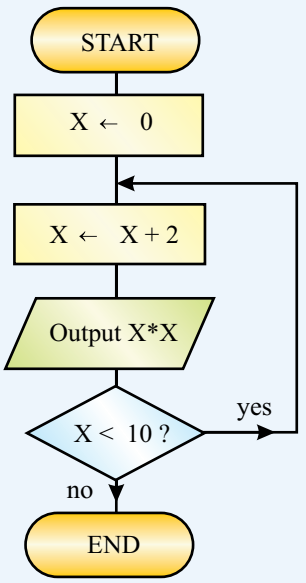
圖 22　計算 2 的 N 次方的總和，N 從 1 到 10

例 15　下列的程式將會顯示在 2 與 10 之間所有偶數的平方，即 2^2、4^2、6^2、8^2、10^2。

```
10  FOR X = 2 TO 10 STEP 2
20      OUTPUT X * X
30  NEXT
```

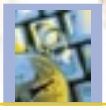
上述程式等同下列的程式：

```
10  X ← 0
20  X ← X + 2
30  OUTPUT X * X
40  IF X < 10 THEN
50      GOTO 20
60  END IF
```



程式的輸出是 4, 16, 36, 64, 100。

圖 23　顯示在 2 與 10 之間偶數的平方



例 16　下列的程式將會輸入十個正數，並找出其中最大的一個。

```
10    MAX ← -1
20    FOR I = 1 TO 10
30        INPUT X
40        IF X > MAX THEN
50            MAX ← X
60        ENDIF
70    NEXT
80    OUTPUT MAX
```

第 10 行將 -1 賦值到變數 MAX。第 40 行將輸入的數據與 MAX 作比較，若輸入的數據較大，輸入的數據將會取代 MAX 先前的內容。

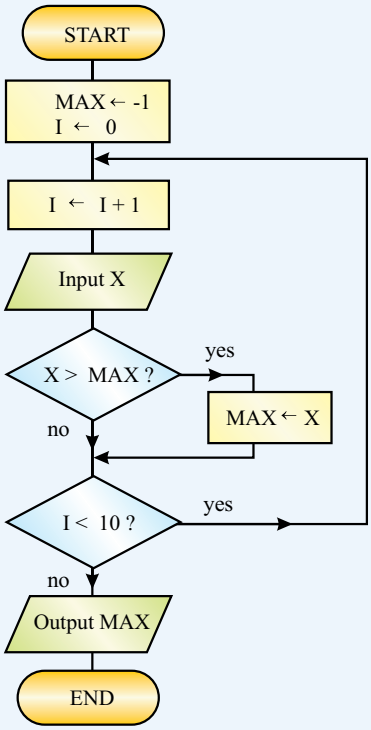


圖 24　計算最大值

16.6　Repeat .. Until 迴路

重　點

Repeat .. Until 迴路將會執行至少一次

Repeat .. Until 重複...直到 迴路的格式是：

```
REPEAT
    {要重複的行動}
UNTIL {條件滿足}
```

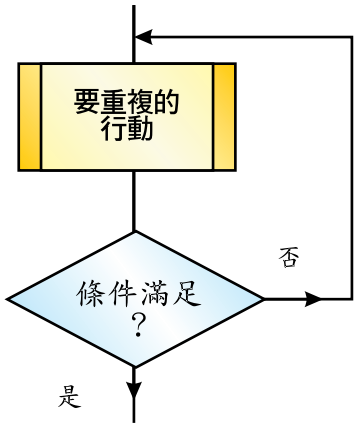


圖 25　Repeat .. Until 迴路

迴路裡的行動將會至少被執行一次。細閱下列的例子：

```
10    REPEAT
20        INPUT X
30    UNTIL X > 0
```

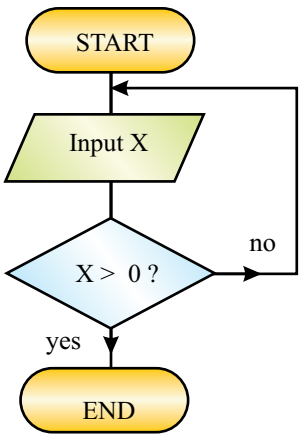


圖 26　確保輸入的數據是正數

上述程式片段首先輸入一個數據，若數據不大於零（即小於或等於零），電腦將要求用戶重新輸入。因此，這程式確保輸入的數據是正數，並等同下列程式：

```
10    INPUT X
20    IF NOT (X > 0) THEN
30        GOTO 10
40    END IF
```


例 17 下列的程式將會找出需要多少個由 1 開始的連續整數，加起來的和至少等於 10。

```
10  S ← 0
20  C ← 0
30  REPEAT
40        C ← C + 1
50        S ← S + C
60  UNTIL S >= 10
80  OUTPUT C
```

在執行第 60 行後	C	S	重複？
第一次	1	1	是
第二次	2	3	是
第三次	3	6	是
第四次	4	10	否

上述程式等同下列的程式：

```
10  S ← 0
20  C ← 0
30  C ← C + 1
40  S ← S + C
50  IF NOT (S >= 10) THEN
60        GOTO 30
70  END IF
80  OUTPUT C
```

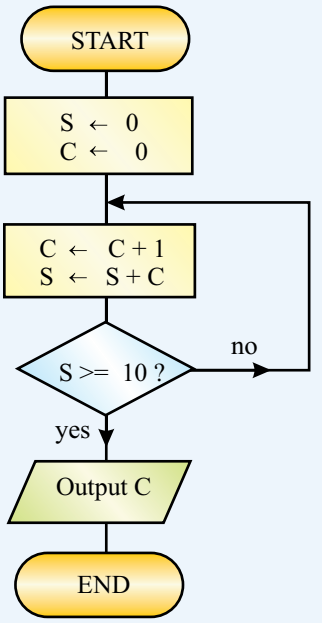


圖 27 Repeat .. Until 迴路的應用

16.7 While 迴路

重 點

While 迴路裡的行動未必一定執行。

While 迴路的格式是

```
WHILE {條件滿足}
      {要重複的行動}
END WHILE
```

迴路裡的行動未必會被執行。

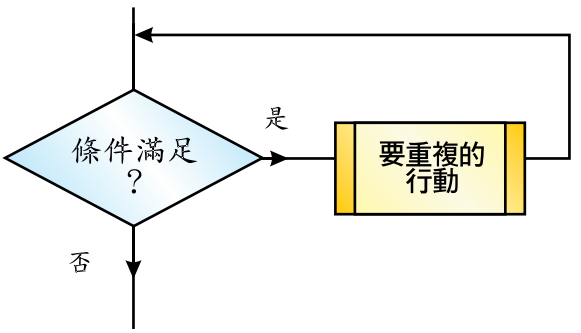


圖 28 迭代控制結構：While迴路

連續整數 consecutive integer

細閱下列的例子：

```
10    INPUT X
20    WHILE X < 0
30        INPUT X
40    END WHILE
```

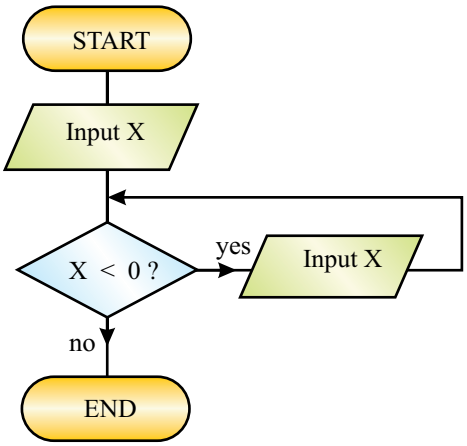


圖 29　確保輸入的數據是正數

上述程式片段將輸入一個數據。若所輸入的值小於零，電腦將會要求用戶重新輸入。因此，這個程式確保輸入的數據不會是負數，並等同下列程式：

```
10    INPUT X
20    IF X < 0 THEN
30        INPUT X
40        GOTO 20
40    END IF
```

例 18　　下列的程式將計算兩個正數相除後的餘數。

```
10    INPUT N
20    INPUT DIVISOR
30    WHILE N >= DIVISOR
40        N ← N - DIVISOR
50    END WHILE
60    OUTPUT N
```

上述程式等同下列的：

```
10    INPUT N
20    INPUT DIVISOR
30    IF N >= DIVISIOR THEN
40        N ← N - DIVISOR
50        GOTO 30
60    END IF
70    OUTPUT N
```

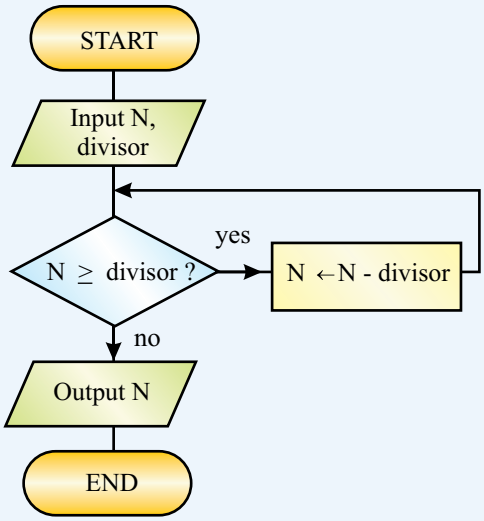


圖 30　使用While 迴路計算餘數

繼續例 18

若被除數 (N) 是 15，而除數 (DIVISOR) 是 4，上述程式會重複三次，每次將 N 減去 4，也就是進行 15 - 4 - 4 - 4，故結果 N = 3 而這就是正確的餘數。另外，若被除數 (N) 是 3，而除數是 4，上述程式並不會執行任何減數，結果得出的餘數亦是 3。

由於 Repeat ... until 迴路至少執行任務一次，若使用 Repeat ... until 迴路來取代上述 While 迴路如下，可能會導致「邏輯錯誤」：

```
10  INPUT N
20  INPUT DIVISOR
30  REPEAT
40        N ← N - DIVISOR
50  UNTIL N < DIVISOR
60  OUTPUT N
```

上述邏輯錯誤發生在當除數比被除數大時，例如 N 是 3 而除數是 4：由於第 40 行必須執行，因此餘數便會錯誤地等於 -1。

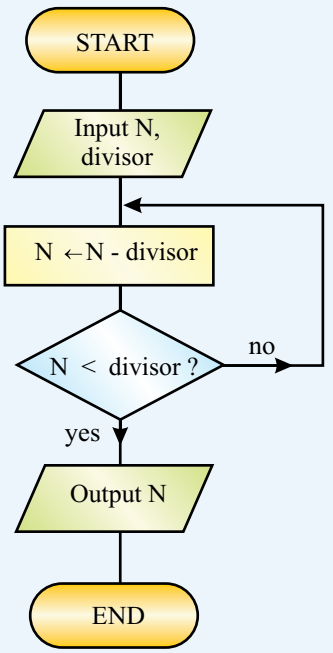
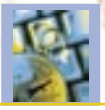


圖 31　若使用 Repeat ... until 迴路來計算餘數，將會導致「邏輯錯誤」

摘要

1. 變數代表一個保存數據的存貯格。賦值語句把數值存貯在變數中，並會重寫它的內容。
2. 電腦首先計算「乘」、「除」，然後才計算「加」、「減」。
3. 輸入語句讓用戶互動式更改變數的值；輸出語句把數據顯示在屏幕上。
4. 若指定的條件得到滿足，條件語句才會執行某個行動。條件算式的結果只有兩種可能：「真」或「假」。關係運算符用於條件算式，包括 =、<>、>、<、>= 和 <=。布爾運算處理多個條件算式，布爾運算符是 AND、 OR 和 NOT。
5. 分支語句將改變程式執行的次序，並以 GOTO 指令實踐。
6. 迭代意謂在指定次數內重複一組指令，迭代結構包括：For 迴路、 Repeat ... until 迴路及 While 迴路。
7. Repeat .. Until 迴路將會執行至少一次； While 迴路則未必。



練習

多項選擇題

1. 下列哪項語句含有語法錯誤？
   (1) x + 1 ← 4
   (2) x ← 5 + 8 * 2
   (3) x ← "5" + 5

   A. 只有 (1)
   B. 只有 (3)
   C. 只有 (1) 和 (3)
   D. 只有 (2) 和 (3)

2. 條件語句包括
   A. IF ... THEN
   B. 賦值語句
   C. GOTO
   D. WHILE

3. 迭代語句包括
   (1) REPEAT .. UNTIL
   (2) WHILE .. END WHILE
   (3) FOR .. NEXT

   A. 只有 (1)
   B. 只有 (3)
   C. 只有 (1) 和 (2)
   D. (1)、(2) 和 (3)

4. 輸入語句
   (1) 必須有至少一個變數
   (2) 必須與輸出語句同時使用
   (3) 不能用於一個 For 迴路中

   A. 只有 (1)
   B. 只有 (3)
   C. 只有 (1) 和 (2)
   D. 只有 (2) 和 (3)

5. 下列哪項的結果是「真」？

   A. (X > X + 5) AND (1 > 2)
   B. (3 > -7) OR (-7 > 3)
   C. (5 >= 5) AND (4 >= 5)
   D. (7+3 < 8) OR (-7 > -6)

6. 細閱下列程式：

```
10    X ← 3
20    Y ← 2
30    Z ← X
40    Y ← Z
50    X ← Z
```

在執行之後，變數的內容將會是

	X	Y	Z
A.	2	2	2
B.	2	3	2
C.	3	2	3
D.	3	3	3

7. 細閱下列的程式：

```
10    INPUT X
20    IF X > 5 THEN
30              X ← 5
40    ENDIF
```

這程式

A. 只允許數字 5 輸入。
B. 只允許小於或等於 5 的數字輸入。
C. 確保 X 的數值大於 5。
D. 確保 X 的數值小於或等於 5。

8. 下列的 For 迴路將會重複執行多少次？

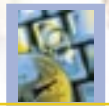
```
100       FOR I = 6 TO 10
  ..
900       NEXT
```

A. 0
B. 4
C. 5
D. 6

9. 下列的 While 迴路將會重複執行多少次？

```
110       I ← 6
120       WHILE I < 10
  ..
900             I ← I + 1
910       END WHILE
```

A. 0
B. 4
C. 5
D. 6



10. 下列的 Repeat ... Until 迴路將會重複執行多少次？

```
110         I ← 6
120         REPEAT
            ..
900               I ← I + 1
910         UNTIL I > 10
```

A. 0
B. 4
C. 5
D. 6

問答題

1. 細閱下列的程式片段：

```
(a)                    (b)                    (c)
10  P ← 1              10  P ← 3              10  P ← -2
20  R ← 8              20  Q ← 4              20  Q ← 3
30  Q ← P + 3          30  R ← P + 5          30  R ← P * (-4)
40  P ← Q              40  P ← P - 2          40  P ← R / 2
50  R ← P              50  Q ← R + 3          50  Q ← Q * (-5)
```

對每個程式片段，計算在執行完成後 P、Q 和 R 的數值。

2. 細閱下列的程式片段：

```
(a)                    (b)                    (c)
10  P ← 6              10  P ← 4              10  P ← 12
20  Q ← 2              20  Q ← 6              20  Q ← 6
30  P ← P + Q          30  P ← P * Q          30  P ← P - Q
40  Q ← Q - P          40  Q ← Q / P          40  Q ← Q * P
50  P ← P + Q          50  P ← P / Q          50  P ← P + Q
```

對每個程式片段，計算在執行完成後 P 和 Q 的數值。

3. 細閱下列的程式片段：

```
(a)                          (b)                          (c)
10  X ← 10                   10  X ← 10                   10  S ← 10
20  S ← 100                  20  S ← 100                  20  S ← S / 2
30  IF X  < 2 THEN           30  IF 2*X  < 15 THEN        30  IF S/2 > 2.5 THEN
40     S ← S - 10            40     S ← S / 10            40     S ← S / 2
50  ELSE                     50  ELSE                     50  ELSE
60     S ← S + 10            60     S ← S * 10            60     S ← S * 2
70  ENDIF                    70  ENDIF                    70  ENDIF
```

對每個程式片段，計算在執行完成後 S 的數值。

4. 填寫下列空格，讓程式合理地執行：

```
10   INPUT X
20   IF _______ THEN
30       OUTPUT "X is positive"
40   ELSE
50       IF _______ THEN
60               OUTPUT "X is zero"
70       ELSE
80               OUTPUT "X is negative"
90       END IF
100  END IF
```

5. 下列的程式是對「應課稅入息實額」X，計算應課繳的薪俸稅 T：

```
120  INPUT X
130  IF X < 35000 THEN
140      T ← X * 0.02
150  ELSE
160      IF X < 70000 THEN
170              T ← 700 + (X - 35000) * 0.07
180      ELSE
190              IF X < 105000 THEN
200                      T ← 3150 + (X - 70000) * 0.12
210              ELSE
220                      T ← 7350 + (X - 105000) * 0.17
230              ENDIF
240      ENDIF
250  END IF
```

對以下各項「應課稅入息實額」，找出應課繳的稅款

(a) 10,000
(b) 50,000
(c) 90,000
(d) 150,000

6. 計算下列每個程式將會打印多少個 1 字：

(a)

```
10   X ← 0
20   OUTPUT 1
30   X ← X + 1
40   IF X <= 5 THEN
50      GOTO 20
60   ENDIF
```

(b)

```
10   X ← 1
20   OUTPUT 1
30   X ← X + 1
40   IF X < 5 THEN
50      GOTO 20
60   ENDIF
```

(c)

```
10   X ← 1
20   IF X < 5 THEN
30      X ← X + 1
40      OUTPUT 1
50      GOTO 20
60   ENDIF
```

7. 寫出下列程式的輸出：

```
10   S ← 100
20   X ← 10
30   S ← S - X
40   X ← X * 2
50   IF S > 0 THEN
60       GOTO 30
70   ELSE
80       OUTPUT S, X
90   END IF
```

8. 寫出下列程式的輸出：

(a)

```
10 X ← 2
20 FOR I = 2 TO 8
30         X ← X + 3
40 NEXT
50 OUTPUT X
```

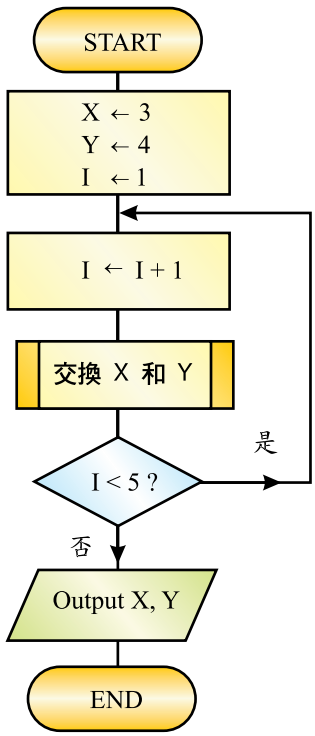
(b)

```
10 X ← 2
20 REPEAT
30         X ← X + 3
40 UNTIL X > 20
50 OUTPUT X
```

(c)

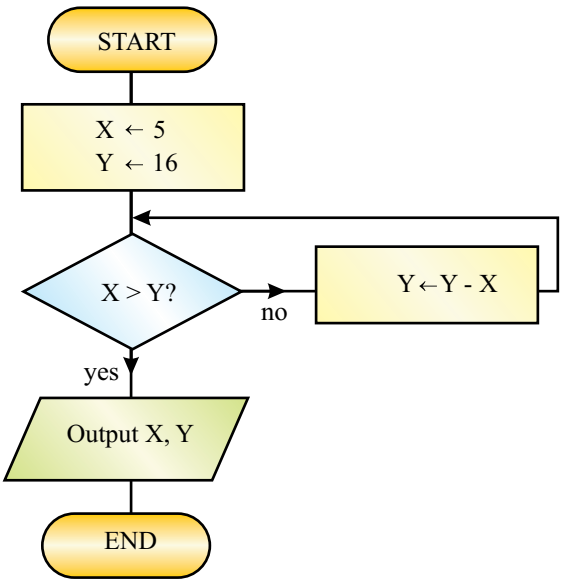
```
10 X ← 2
20 WHILE X < 20
30         X ← X + 3
40 END WHILE
50 OUTPUT X
```

9. 細閱右邊的程式流程圖：
   (a) 為程序 "交換 X 和 Y" 編寫一個程式片段。
   (b) 寫出流程圖的輸出。

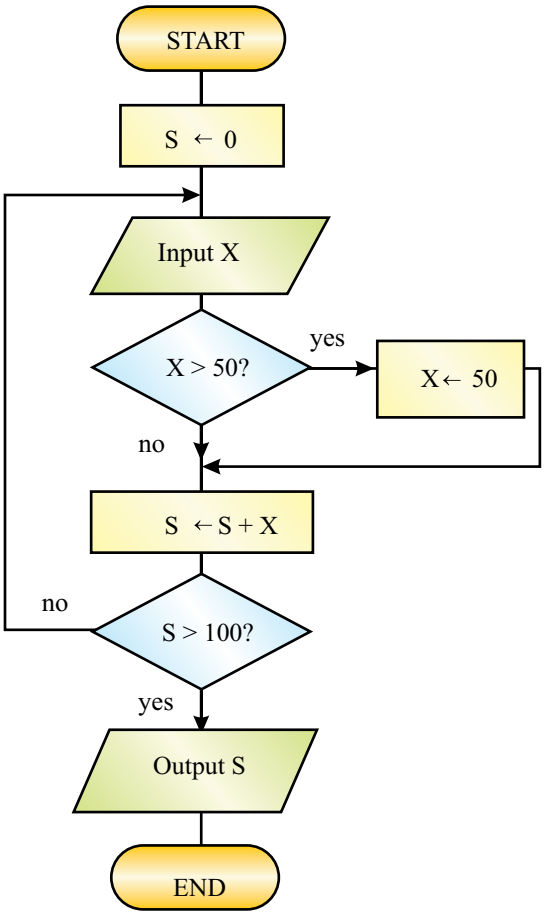


10. 編寫一個程式來計算從鍵盤輸入十個數字中最小的值。假設所有數字都小於 100 。

11. 寫出下列程式流程圖的輸出：



12. 細閱下列的程式流程圖：



為下列各項輸入，寫出這個程式的輸出：

(a) 10, 20, 60, 10, 20

(b) 120, 130, -2, 200