流程控制

人工智慧與無線感應設備開發專班

湜憶電腦知訊顧問股份有限公司

馬傳義

前言

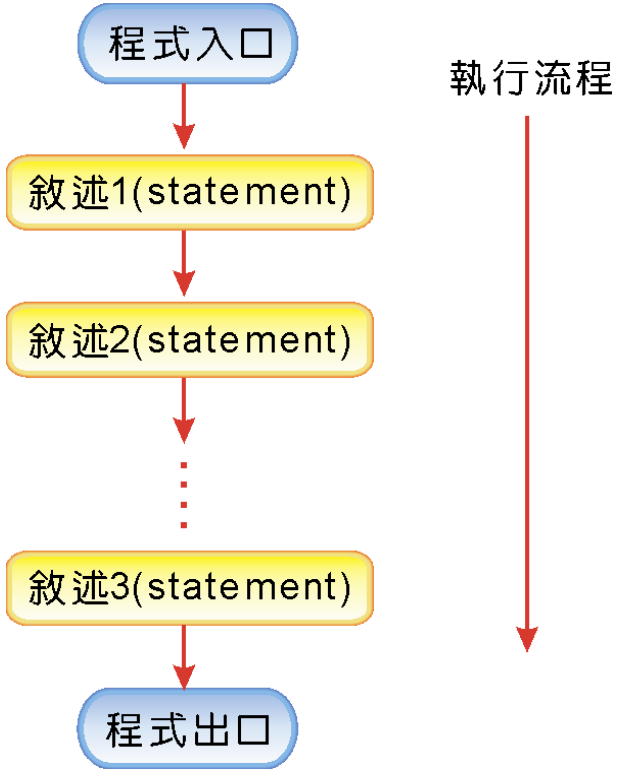
- 任何一種程式語言，最基本的流程控制都是由「循序結構」、「選擇結構」、「重複結構」和「分支結構」四者組合而成的程式碼。
  - 「循序結構」的特性是從頭到尾、自上而下，一個敘述接著一個敘述逐行執行。
  - 「選擇結構」是當程式執行遇到分歧時，流程要往哪個敘述區段（statements）走，就要視當時資料所符合的條件來決定。
  - 「重複結構」是程式中有某個敘述區段需要被重複執行時使用，能否被重複執行也是根據當時資料所符合的條件來決定。
  - 「分支結構」則可以控制程式執行的順序。

前言

- 只要擁有清楚的邏輯分析能力，並配合以上四種結構，就能寫出一個良好結構化的程式出來。

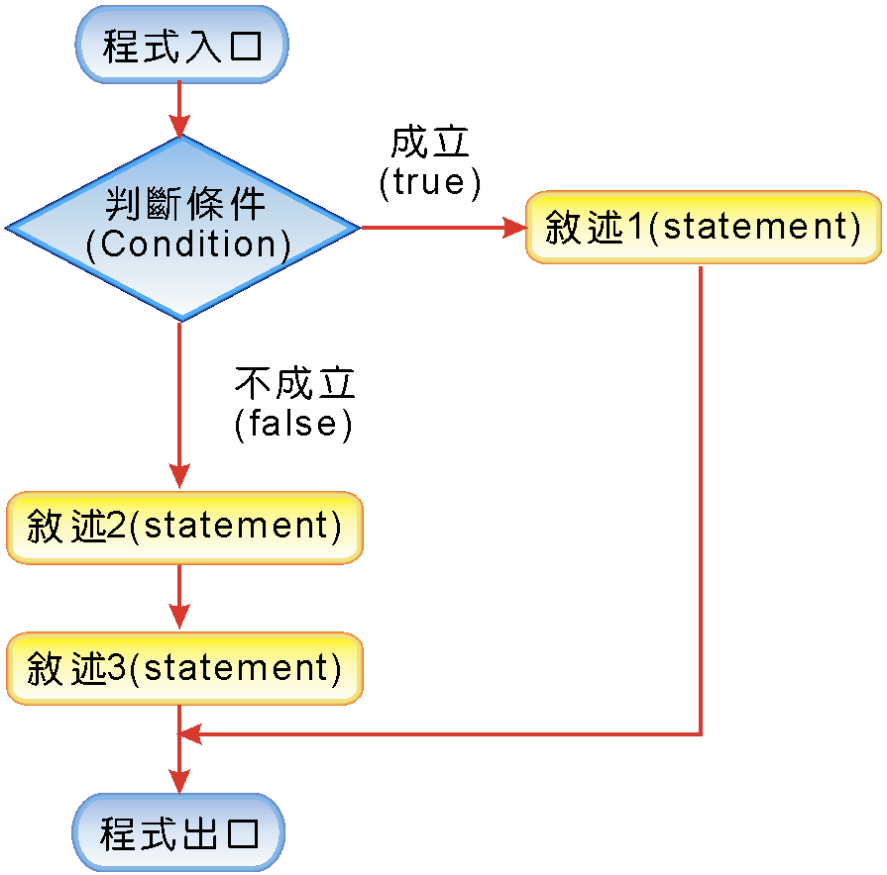
循序結構

- 循序式結構在整個程式，沒有特殊的流程、分支或跳動、大部份的程式都是依照此結構模組（Modules）來設計。



選擇結構

- 選擇式結構有個重點，不論條件成立或不成立，最後都是由同一出口，結束流程。



選擇結構

- 單向選擇（if）
  - 語法：

```
if (條件判斷)
{
   程式敘述區;
}
```

    - 說明：
      - 當條件判斷成立（True）時，才會去執行程式敘述區段；若條件判斷不成立（False），該程式敘述區段就不會被執行。

      注意：

      若「程式敘述區」中，只有一行程式，大括號可以省略。

選擇結構

- 程式：

```
public class CH03_01
{
public static void main(String[] args)
    {
        int Tim = 20, Tracy = 23;
        System.out.println("Tim年齡=" + Tim + ",Tracy年齡=" + Tracy);

        if (Tim < Tracy)
            System.out.println("Tim年齡比Tracy小");

        if (Tim > Tracy)
            System.out.println("Tim年齡比Tracy大");

        System.out.println("程式結束");
    }
}
```

選擇結構

```
1  public class CH03_01
2  {
3     public static void main(String[] args)
4     {
5        int Tim = 20, Tracy = 23;
6        System.out.println("Tim年齡=" + Tim + ",Tracy年齡=" + Tracy);
7
8        if (Tim < Tracy)
9           System.out.println("Tim年齡比Tracy小");
10
11       if (Tim > Tracy)
12          System.out.println("Tim年齡比Tracy大");
13
14       System.out.println("程式結束");
15    }
16 }
```

- 執行結果：

```
Tim年齡=20,Tracy年齡=23
Tim年齡比Tracy小
程式結束
```

選擇結構

- 說明：
  - 行05：
    - 宣告變數，並指定初值。
  - 行06：
    - 螢幕輸出。
  - 行08~行09：
    - 單向選擇。
      - 行08：

        條件判斷式若條件成立（即 結果為True），則執行大括號內的敘述；若條件不成立（即 結果為False），則跳過大括號內的敘述。
      - 行09：

        螢幕輸出。

選擇結構

- 行11~行12：
  - 單向選擇。
    - 行11：

      條件判斷式若條件成立（即 結果為True），則執行大括號內的敘述；若條件不成立（即 結果為False），則跳過大括號內的敘述。
    - 行12：

      螢幕輸出。
- 行14：
  - 螢幕輸出。

選擇結構

- 雙向選擇（if … else）
  - 語法：

```
if (條件判斷)
{
    程式敘述區 (1);
}
else
{
    程式敘述區 (2);
}
```

選擇結構

- 說明：
  - 當條件判斷成立（True）時，會執行程式敘述區（1）；
    若條件判斷不成立（False），會執行程式敘述區（2）。

注意：

若「程式敘述區」中，只有一行程式，大括號可以省略。

選擇結構

- 程式：

```
public class CH03_02
{
    public static void main(String[] args)
    {
        int Tim = 25, Tracy = 23;
        System.out.println("Tim年齡=" + Tim + ",Tracy年齡=" + Tracy);

        if (Tim < Tracy)
            System.out.println("Tim年齡比Tracy小");
        else
            System.out.println("Tim年齡比Tracy大");

        System.out.println("程式結束");
    }
}
```

選擇結構

```
1  public class CH03_02
2  {
3     public static void main(String[] args)
4     {
5        int Tim = 25, Tracy = 23;
6        System.out.println("Tim年齡=" + Tim + ",Tracy年齡=" + Tracy);
7
8        if (Tim < Tracy)
9           System.out.println("Tim年齡比Tracy小");
10       else
11          System.out.println("Tim年齡比Tracy大");
12
13       System.out.println("程式結束");
14    }
15 }
```

- 執行結果：

```
Tim年齡=25,Tracy年齡=23
Tim年齡比Tracy大
程式結束
```

選擇結構

- 說明：
  - 行05：
    - 宣告變數，並指定初值。
  - 行06：
    - 螢幕輸出。
  - 行08~行11：
    - 雙向選擇。
      - 行08：

        條件判斷式若條件成立（即 結果為True），則執行行09的敘述；若條件不成立（即 結果為False），則執行行11的敘述。
      - 行09：

        螢幕輸出。
      - 行11：

        螢幕輸出。

選擇結構

- 行13：
  - 螢幕輸出。

選擇結構

- 條件運算子（?:）
  - 是一個「三元運算子（Ternary Operator）」。
  - 簡單的雙向選擇可由條件運算子「?:」來取代。
    - 條件運算子會根據條件式的布林值，從指定的兩個資料之中傳回其中的一個，而待選的資料可以是任何型別的資料、變數或運算式。
    - 它和if-else條件敘述功能一樣，不過這裡的程式敘述只允許單行運算式。
    - 可以用來替代簡單的if-else條件敘述，讓程式碼看起來更為簡潔。
  - 語法：

變數=條件運算式?程式敘述區（1）:程式敘述區（2）;

選擇結構

- 程式：

```
public class CH03_03
{
    public static void main(String[] args)
    {
        int Tim = 25, Tracy = 23;
        String str;

        System.out.println("Tim年齡=" + Tim + ",Tracy年齡=" + Tracy);

        str = (Tim < Tracy) ? "Tim年齡比Tracy小" : "Tim年齡比Tracy大";

        System.out.println(str);
        System.out.println("程式結束");
    }
}
```

選擇結構

```
1  public class CH03_03
2  {
3     public static void main(String[] args)
4     {
5        int Tim = 25, Tracy = 23;
6        String str;
7
8        System.out.println("Tim年齡=" + Tim + ",Tracy年齡=" + Tracy);
9
10       str = (Tim < Tracy) ? "Tim年齡比Tracy小" : "Tim年齡比Tracy大";
11
12       System.out.println(str);
13       System.out.println("程式結束");
14    }
15 }
```

- 執行結果：

```
Tim年齡=25,Tracy年齡=23
Tim年齡比Tracy大
程式結束
```

選擇結構

- 說明：
  - 行05~行06：
    - 宣告變數，並指定初值。
  - 行08：
    - 螢幕輸出。
  - 行10：
    - 條件運算子。
      - 若條件成立（即 結果為True），將問號後、冒號前字串，存入變數str中；若條件不成立（即 結果為False），則將冒號後字串，存入變數str中。
  - 行12~行13：
    - 螢幕輸出。

選擇結構

- 巢狀選擇
  - 如果在選擇結構的內層又有選擇結構，就形成了巢狀選擇結構。
  - 當條件式中有超過三個選擇項時，可以使用巢狀選擇結構。
  - 內層可以是if敘述或者是if-else敘述。
  - 語法：

```
if (主條件)
{
   if (次條件1)
   {
      敘述區段A;
   }
  [ else ]
```

選擇結構

```
    [ { ]
        [ 敘述區段B; ]
    [ } ]
[ } ]
[ else ]
[ { ]
    [ if (次條件2) ]
    [ { ]
        [ 敘述區段C; ]
    [ } ]
    [ else ]
    [ { ]
        [ 敘述區段D; ]
    [ } ]
}
```

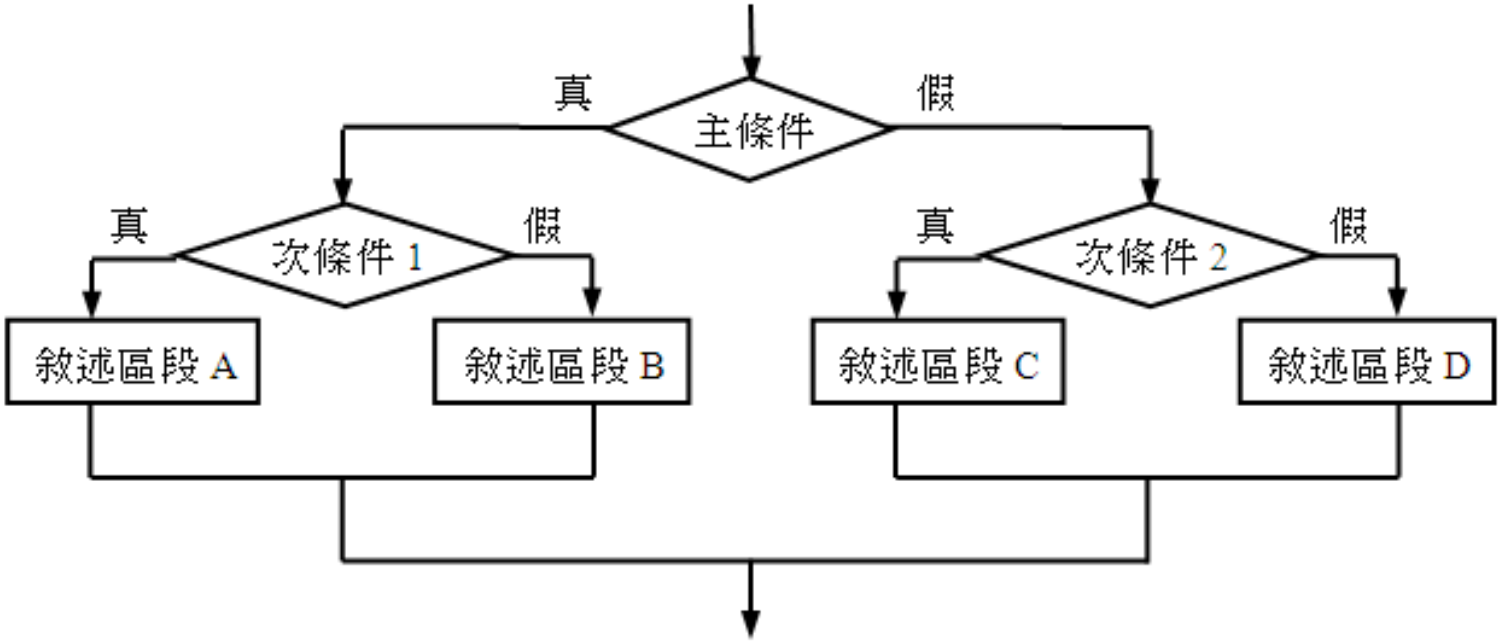
選擇結構

- 說明：
  - 當「主條件」成立時，
    - 則判斷「次條件1」，
      - 「次條件1」成立時執行「敘述區段A」。
      - 不成立則執行「敘述區段B」（若無，則省略）。
  - 若「主條件」不成立（若無，則省略），
    - 則判斷「次條件2」（若無，則省略），
      - 「次條件2」成立時會執行「敘述區段C」（若無，則省略）。
      - 不成立則執行「敘述區段D」（若無，則省略）。

選擇結構

- 說明：
  - 流程圖：

    下圖是最複雜的巢狀選擇流程架構，尚有其它變化的流程架構。



選擇結構

- 程式：

```
import java.io.*;

public class CH03_04
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader keyin;
        keyin = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("請輸入0或1：");
        String st1 = keyin.readLine();
        int num1 = Integer.parseInt(st1);

        System.out.print("請輸入0或1：");
        String st2 = keyin.readLine();
        int num2 = Integer.parseInt(st2);
```

選擇結構

```
        System.out.print("\n" + "AND 邏輯閘 ( " + num1 + " , " + num2 + " )");

        if (num1 == 0)
        {
            if (num2 == 0)
                System.out.println(" = 0");
            else
                System.out.println(" = 0");
        }
        else
        {
            if (num2 == 0)
                System.out.println(" = 0");
            else
                System.out.println(" = 1");
        }
    }
}
```

選擇結構

```
1  import java.io.*;
2
3  public class CH03_04
4  {
5     public static void main(String[] args) throws IOException
6     {
7        BufferedReader keyin;
8        keyin = new BufferedReader(new InputStreamReader(System.in));
9        System.out.print("請輸入0或1 : ");
10       String st1 = keyin.readLine();
11       int num1 = Integer.parseInt(st1);
12
13       System.out.print("請輸入0或1 : ");
14       String st2 = keyin.readLine();
15       int num2 = Integer.parseInt(st2);
16
17       System.out.print("\n" + "AND 邏輯閘 ( " + num1 + " , " + num2 + " )");
18
19       if (num1 == 0)
20       {
21          if (num2 == 0)
22             System.out.println(" = 0");
23          else
24             System.out.println(" = 0");
25       }
26       else
```

選擇結構

```
27      {
28          if (num2 == 0)
29              System.out.println(" = 0");
30          else
31              System.out.println(" = 1");
32      }
33  }
34  }
```

- 執行結果：

```
請輸入0或1：1
請輸入0或1：1

AND 邏輯閘(1,1)=1
```

- 說明：
  - 行01：
    - 載入Java.io.*套件。

選擇結構

- 行05：
  - main()方法後面，加上 throws IOException。
- 行07：
  - 宣告BufferedReader類別的物件。
- 行08：
  - 建立keyin物件。
- 行09：
  - 螢幕輸出。
- 行10~行11：
  - 輸入字串，並將輸入的字串，轉換成整數型態，再指定給變數num1。
- 行13：
  - 螢幕輸出。

選擇結構

- 行14~行15：
  - 輸入字串，並將輸入的字串，轉換成整數型態，再指定給變數num2。
- 行17：
  - 螢幕輸出。
- 行19~行32：
  - 巢狀選擇及螢幕輸出。

選擇結構

- 課堂練習：
  - 請將此（CH03_04）程式，重新改寫為『OR邏輯閘判斷』。

選擇結構

- 多重選擇（if … else if … else）
  - 當選擇的項目超過兩個時，除了可以用巢狀選擇結構來解決外，在特殊情況下，可簡化使用else if多重選擇結構來處理。
  - 語法：

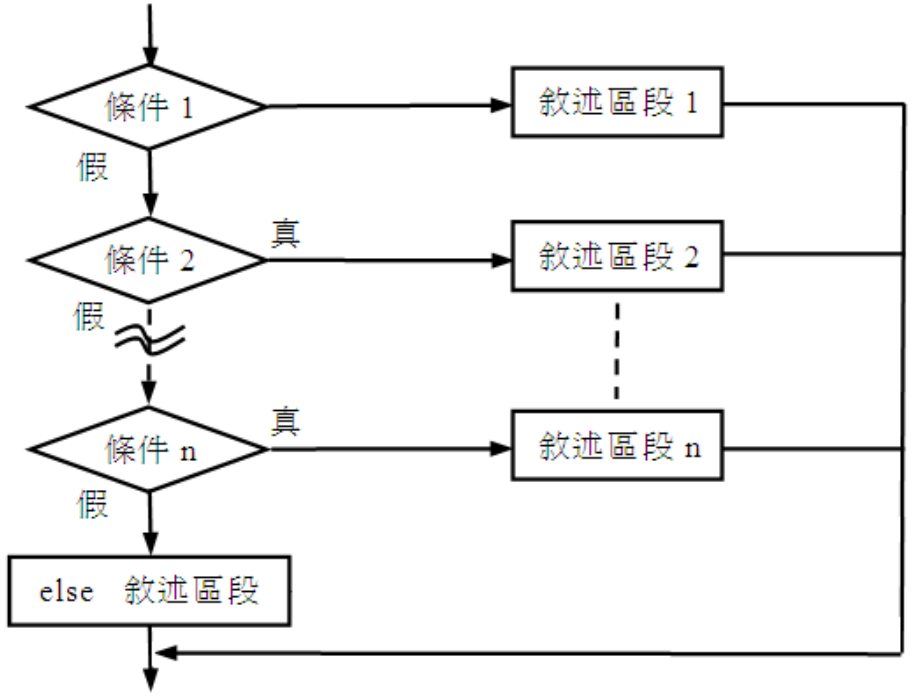
```
if (條件1)
{
   敘述區段1;
}
else if (條件2)
{
   敘述區段2;
```

選擇結構

```
}
      ⋮
else if (條件n)
{
   敘述區段n;
}
else
{
   敘述區段
}
```

選擇結構

- 說明：
  - 若（條件1）成立，則執行敘述區段1；若（條件2）成立，則執行敘述區段2，以此類推，若都不符合條件，則執行else敘述區段。
  - 流程圖：



選擇結構

- 程式：

```
import java.io.*;

public class CH03_05
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader keyin;
        keyin = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("請輸入0或1：");
        String st1 = keyin.readLine();
        int num1 = Integer.parseInt(st1);

        System.out.print("請輸入0或1：");
        String st2 = keyin.readLine();
        int num2 = Integer.parseInt(st2);
```

選擇結構

```
        System.out.print("\n" + "AND 邏輯閘 ( " + num1 + " , " + num2 + " )");
        if ((num1 == 0) && (num2 == 0))
            System.out.println(" = 0");
        else if ((num1 == 0) && (num2 == 1))
            System.out.println(" = 0");
        else if ((num1 == 1) && (num2 == 0))
            System.out.println(" = 0");
        else
            System.out.println(" = 1");
    }
}
```

選擇結構

```
1  import java.io.*;
2
3  public class CH03_05
4  {
5     public static void main(String[] args) throws IOException
6     {
7        BufferedReader keyin;
8        keyin = new BufferedReader(new InputStreamReader(System.in));
9        System.out.print("請輸入0或1：");
10       String st1 = keyin.readLine();
11       int num1 = Integer.parseInt(st1);
12
13       System.out.print("請輸入0或1：");
14       String st2 = keyin.readLine();
15       int num2 = Integer.parseInt(st2);
16
17       System.out.print("\n" + "AND 邏輯閘 ( " + num1 + " , " + num2 + " )");
18
19       if ((num1 == 0) && (num2 == 0))
20          System.out.println(" = 0");
21       else if ((num1 == 0) && (num2 == 1))
22          System.out.println(" = 0");
23       else if ((num1 == 1) && (num2 == 0))
24          System.out.println(" = 0");
25       else
26          System.out.println(" = 1");
```

選擇結構

```
27    }
28 }
```

- 執行結果：

```
請輸入0或1：1
請輸入0或1：1

AND 邏輯閘(1,1)=1
```

- 說明：
  - 行01：
    - 載入Java.io.*套件。
  - 行05：
    - main()方法後面，加上 throws IOException。
  - 行07：
    - 宣告BufferedReader類別的物件。

選擇結構

- 行08：
  - 建立keyin物件。
- 行09：
  - 螢幕輸出。
- 行10~行11：
  - 輸入字串，並將輸入的字串，轉換成整數型態，再指定給變數num1。
- 行13：
  - 螢幕輸出。
- 行14~行15：
  - 輸入字串，並將輸入的字串，轉換成整數型態，再指定給變數num2。
- 行17：
  - 螢幕輸出。

選擇結構

- 行19~行26：
  - 多重選擇及螢幕輸出。

選擇結構

- 條件選擇（switch）
  - 當選擇的項目超過兩個以上時，除了使用if … else if … else的多重選擇結構外，還可以考慮使用switch選擇結構。
  - 若能使用switch選擇結構，則程式敘述比較簡潔易讀。
  - 語法：

```
switch (運算式或變數)
{
    case數值1:
        敘述區段1;
        break;
```

選擇結構

```
    case數值2:
        敘述區段2;
        break;
          ⋮
    case結果n:
        敘述區段n;
        break;
    default:
        敘述區段;
}
```

選擇結構

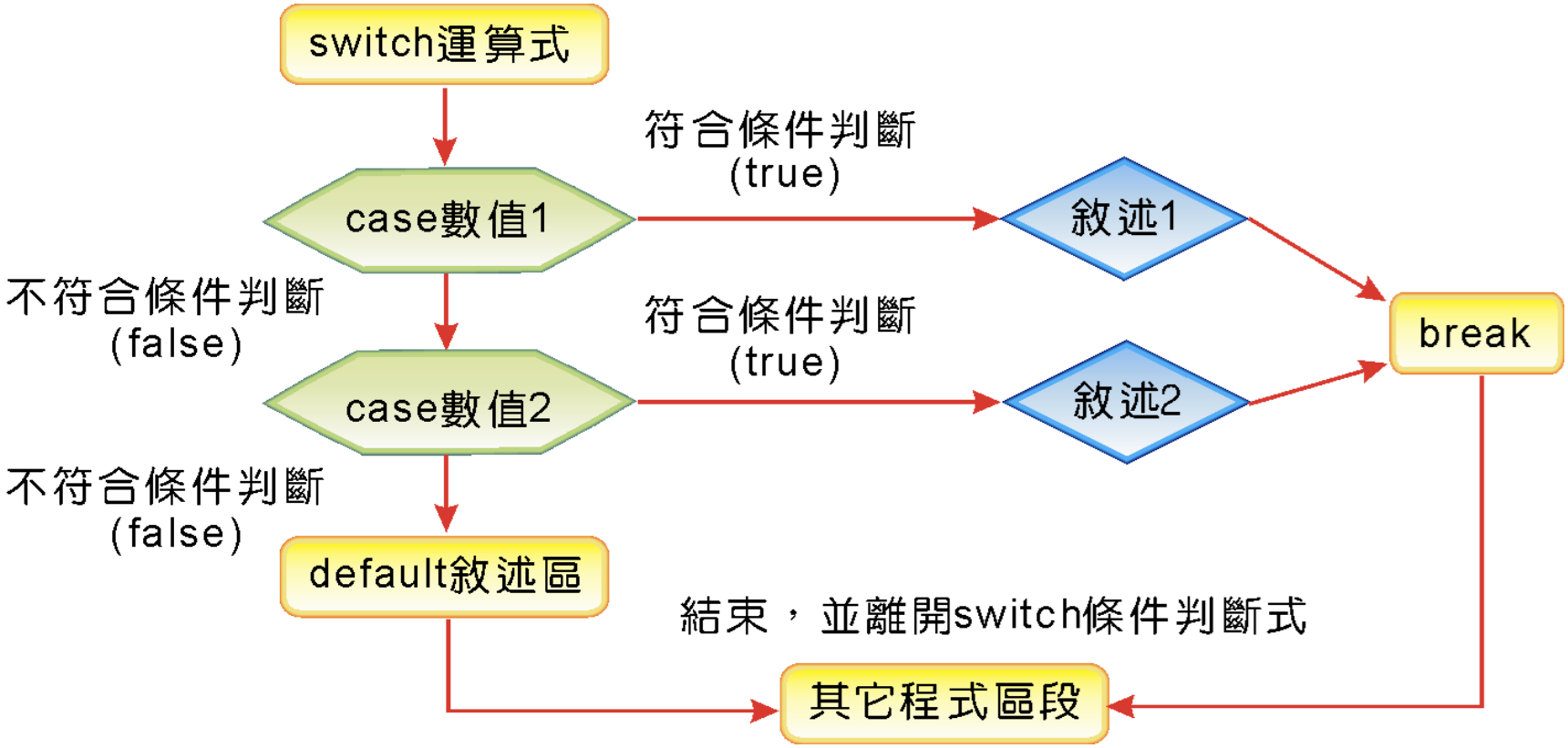
- 說明：
  - 依據不同的運算式結果或變數值，執行對應case內的敘述區段。
    - 若符合數值1，則執行敘述區段1；符合數值2，則執行敘述區段2 …以此類推；都不符合時，則執行default敘述區段。

注意：

- 在switch選擇結構的每一個case的敘述區段後面，須加上跳離指令break，才會在執行完該區段後，跳離開switch選擇結構，否則在執行了符合的case敘述區段之後，會接著執行下一個case敘述區段。
- 在switch選擇結構的最後面，一定要有default敘述區段，以避免碰到前面case值都不滿足時所產生的問題。

選擇結構

- 流程圖：


switch運算式
case數值1
符合條件判斷
(true)
敘述1
不符合條件判斷
(false)
case數值2
符合條件判斷
(true)
敘述2
break
不符合條件判斷
(false)
default敘述區
結束，並離開switch條件判斷式
其它程式區段

選擇結構

- 程式：

```
public class CH03_06
{
    public static void main(String[] args)
    {
        char math_score = 'A';
        System.out.println("Michael數學成績：" + math_score);
        switch (math_score)
        {
        case 'A':
            System.out.println("師長評語：非常好！真是優秀" + "\n");
            break;
        case 'B':
            System.out.println("師長評語：也不錯，但還可以更好" + "\n");
            break;
        case 'C':
            System.out.println("師長評語：真的要多用功" + "\n");
            break;
```

選擇結構

```
default:
    System.out.println("師長評語：不要貪玩，為自己多讀書" + "\n");
}

math_score = 'C';
System.out.println("Jane數學成績：" + math_score);
switch (math_score)
{
case 'A':
    System.out.println("師長評語：非常好！真是優秀" + "\n");
    break;
case 'B':
    System.out.println("師長評語：也不錯，但還可以更好" + "\n");
    break;
case 'C':
    System.out.println("師長評語：真的要多用功" + "\n");
    break;
default:
    System.out.println("師長評語：不要貪玩，為自己多讀書" + "\n");
```

選擇結構

```
            }
        }
    }
```

選擇結構

```
1  public class CH03_06
2  {
3     public static void main(String[] args)
4     {
5        char math_score = 'A';
6        System.out.println("Michael數學成績：" + math_score);
7        switch (math_score)
8        {
9        case 'A':
10          System.out.println("師長評語：非常好！真是優秀" + "\n");
11          break;
12       case 'B':
13          System.out.println("師長評語：也不錯，但還可以更好" + "\n");
14          break;
15       case 'C':
16          System.out.println("師長評語：真的要多用功" + "\n");
17          break;
18       default:
19          System.out.println("師長評語：不要貪玩，為自己多讀書" + "\n");
20       }
21
22       math_score = 'C';
23       System.out.println("Jane數學成績：" + math_score);
24       switch (math_score)
25       {
26       case 'A':
```

選擇結構

```
27            System.out.println("師長評語：非常好！真是優秀" + "\n");
28            break;
29         case 'B':
30            System.out.println("師長評語：也不錯，但還可以更好" + "\n");
31            break;
32         case 'C':
33            System.out.println("師長評語：真的要多用功" + "\n");
34            break;
35         default:
36            System.out.println("師長評語：不要貪玩，為自己多讀書" + "\n");
37         }
38      }
39   }
```

- 執行結果：

```
Michael數學成績：A
師長評語：非常好！真是優秀

Jane數學成績：C
師長評語：真的要多用功
```

選擇結構

- 說明：
  - 行05：
    - 設定數學成績變數（math_score），並指定初始值。
  - 行06：
    - 螢幕輸出。
  - 行07~行20：
    - 設定使用switch選擇結構。
    - 行09~行11：
      - 若變數（math_score）內容為'A'，則執行此區段內螢幕輸出（行10）及跳出switch選擇結構（行11）的敘述。
    - 行12~行14：
      - 若變數（math_score）內容為'B'，則執行此區段內螢幕輸出（行13）及跳出switch選擇結構（行14）的敘述。

選擇結構

- 行15~行17：
  - 若變數（math_score）內容為'C'，則執行此區段內螢幕輸出（行16）及跳出switch選擇結構（行17）的敘述。
- 行18~ 19：
  - 若變數（math_score）內容不是'A'、'B'或'C'，則執行此預設（default）區段內螢幕輸出（行19），並離開此switch選擇結構。

- 行22：
  - 重新設定數學成績變數（math_score）的值。
- 行23：
  - 螢幕輸出。
- 行24~行37：
  - 設定使用switch選擇結構。

選擇結構

- 行26~行28：
  - 若變數（math_score）內容為'A'，則執行此區段內螢幕輸出（行27）及跳出switch選擇結構（行28）的敘述。
- 行29~行31：
  - 若變數（math_score）內容為'B'，則執行此區段內螢幕輸出（行30）及跳出switch選擇結構（行31）的敘述。
- 行32~行34：
  - 若變數（math_score）內容為'B'，則執行此區段內螢幕輸出（行33）及跳出switch選擇結構（行34）的敘述。
- 行35~行36：
  - 若變數（math_score）內容不是'A'、'B'或'C'，則執行此預設（default）區段內螢幕輸出（行36）的敘述，並離開此switch選擇結構。

選擇結構

註：

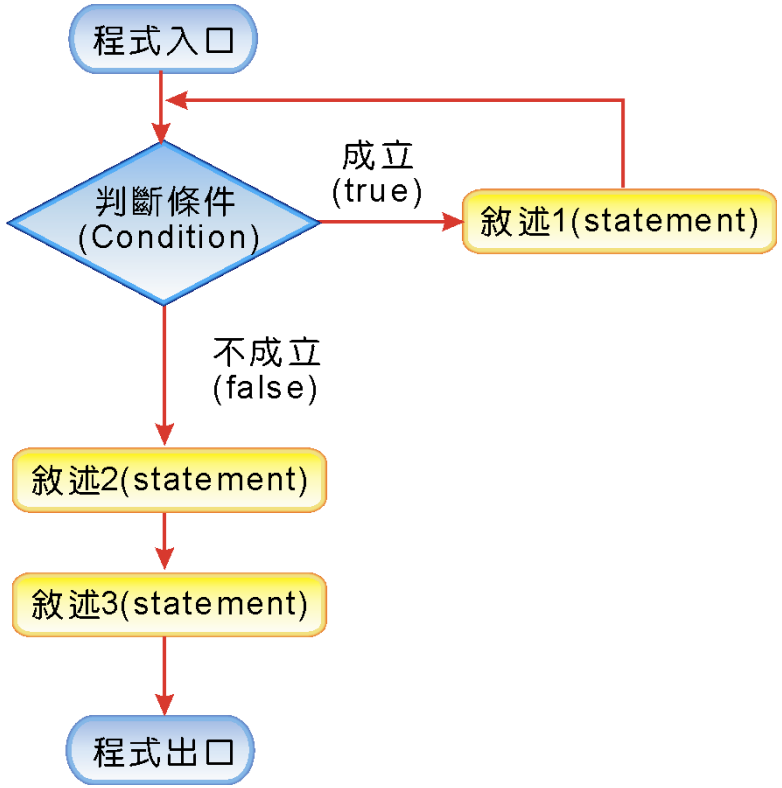
每一個case結束都會有加上break，目的就是，如果已經滿足該case的條件，其餘的case就不需要再進行比對，可以直接離開switch條件式。

選擇結構

- 課堂練習：
  - 請將此（CH03_06）程式，用已知的『輸入』指令，重新改寫（即 math_score變數，改由使用者輸入，再由程式判斷等第）。

重複結構

- 重複結構是一種迴圈控制，根據所設立的條件，重複執行某一段程式敘述，直到條件判斷不成立，才會跳出迴圈。



重複結構

- for迴圈
  - 語法：

    for (起始值;判斷條件;遞增值)

    {

    程式敘述區;

    }
  - 說明：
    - 起始值：
      - 是for迴圈第一次開始的條件數值。
    - 判斷條件：
      - 當for敘述迴圈的判斷條件結果為false時，迴圈就會結束。

重複結構

- 遞增或遞減算式：
  - 每次執行迴圈後，起始值要增加或減少的算式。

- 執行步驟：

1.設定控制變數的起始值。

2.如果條件運算式為真則執行for迴圈內的敘述；如果條件運算式為假，則跳離for迴圈。

3.執行完成之後，視使用者的需求來增加或減少控制變數的值，再重複步驟2。

例如：

```
for ( int i=0;i <=5;i++ )
{
   a=a+5;
}
```

重複結構

- 程式：

```
public class CH03_07
{
    public static void main(String[] args)
    {
        System.out.println("1~10間奇數的和");

        int sum = 0;

        System.out.println("所有的奇數：");

        for (int i = 1; i <= 10; i++)
            if (i % 2 != 0)
            {
                sum += i;
                System.out.print(i + " ");
            }

        System.out.println();
```

重複結構

```
            System.out.print("答案=" + sum);
        }
    }
```

重複結構

```
1  public class CH03_07
2  {
3     public static void main(String[] args)
4     {
5        System.out.println("1~10間奇數的和");
6
7        int sum = 0;
8
9        System.out.println("所有的奇數：");
10
11       for (int i = 1; i <= 10; i++)
12          if (i % 2 != 0)
13          {
14             sum += i;
15             System.out.print(i + " ");
16          }
17
18       System.out.println();
19       System.out.print("答案=" + sum);
20    }
21 }
```

重複結構

- 執行結果：

```
1~10間奇數的和
所有的奇數：
1 3 5 7 9
答案=25
```

- 說明：
  - 行05：
    - 螢幕輸出。
  - 行07：
    - 宣告整數變數。
  - 行09：
    - 螢幕輸出。

重複結構

- 行11~行16：
  - 計算1~10間奇數的總和。
  - 顯示1~10間的奇數數字。
  - 行11：
    - 設定for迴圈。

      包含設定迴圈的起始值（即 int i = 1）、判斷條件（即 i <= 10）及遞增值（即 i++）。

      因為控制變數i從1開始，一直執行到i=11時，才會因為不符合判斷條件而離開，故此迴圈（行11~行16）共要執行10次，且迴圈執行完畢時，i會等於11。
  - 行12~行16：
    - 利用i%2的結果判斷i是否為奇數。

      當i%2的結果不為0時，即表示i為奇數。
    - 行14：

      計算奇數的和。

重複結構

註：

sum += i為sum = sum + i的精簡寫法。

- 行15：

  螢幕輸出奇數的數字。

- 行18：
  - 螢幕輸出一行（即 換行）。
- 行19：
  - 螢幕輸出奇數的和。

重複結構

- 課堂練習：
  - 請將此（CH03_07）程式，重新改寫為『印出1~10間的偶數，並計算出1~10間偶數的和』。

重複結構

- 巢狀for迴圈
  - 語法：

```
for ( 起始值;判斷條件;遞增值 )
{
        ⋮
    for ( 起始值;判斷條件;遞增值 )
    {
        程式敘述區;
     }
        ⋮
}
```

重複結構

- 程式：

```
public class CH03_08
{
    public static void main(String[] ages)
    {
        for (int i = 1; i <= 9; i++)
        {
            for (int j = 1; j <= 9; j++)
                System.out.print(i + "*" + j + "=" + i * j + '\t');

            System.out.print('\n');
        }
    }
}
```

重複結構

```
1  public class CH03_08
2  {
3     public static void main(String[] ages)
4     {
5        for (int i = 1; i <= 9; i++)
6        {
7           for (int j = 1; j <= 9; j++)
8              System.out.print(i + "*" + j + "=" + i * j + '\t');
9
10          System.out.print('\n');
11       }
12    }
13 }
```

- 執行結果：

```
1*1=1   1*2=2   1*3=3   1*4=4   1*5=5   1*6=6   1*7=7   1*8=8   1*9=9
2*1=2   2*2=4   2*3=6   2*4=8   2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*1=3   3*2=6   3*3=9   3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*1=4   4*2=8   4*3=12  4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*1=5   5*2=10  5*3=15  5*4=20  5*5=25  5*6=30  5*7=35  5*8=40  5*9=45
6*1=6   6*2=12  6*3=18  6*4=24  6*5=30  6*6=36  6*7=42  6*8=48  6*9=54
7*1=7   7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49  7*8=56  7*9=63
8*1=8   8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64  8*9=72
9*1=9   9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

重複結構

- 說明：
  - 行05~行11：
    - 印出99乘法表（橫式）。
    - 行05：
      - 設定外層for迴圈。

        包含設定迴圈的起始值（即 int i = 1）、判斷條件（即 i <= 9）及遞增值（即 i++）。

        因為控制變數i從1開始，一直執行到i=10時，才會因為不符合判斷條件而離開，故此迴圈（行05~行12）共要執行9次，且迴圈執行完畢時，i會等於10。
      - 行07：

        設定內層for迴圈。

        包含設定迴圈的起始值（即 int j = 1）、判斷條件（即 j <= 9）及遞增值（即 j++）。

重複結構

因為控制變數j從1開始，一直執行到j=10時，才會因為不符合判斷條件而離開，故此迴圈（行07~行10）共要執行9次，且迴圈執行完畢時，j會等於10。

行08：

螢幕輸出。

先從螢幕輸出「i*j=」及i*j的「值」，再利用逃逸字元的「\t」，讓游標移到下一個水平定位。

- 行10：
  - 螢幕輸出一行（即換行）。

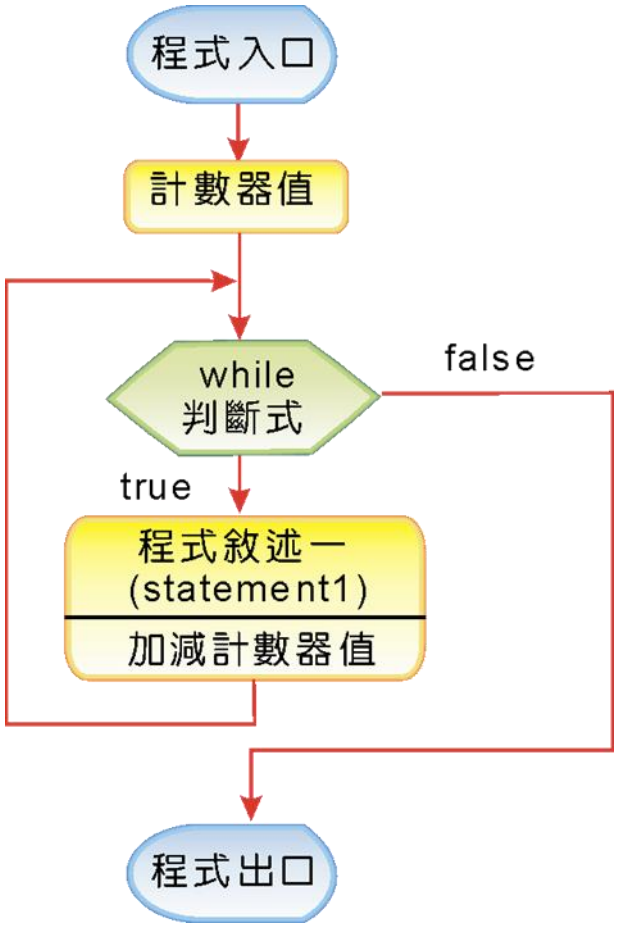
重複結構

- 課堂練習：
  - 請將此（CH03_08）程式，重新改寫為『直印』。
  - 結果：

```
1*1=1  2*1=2  3*1=3  4*1=4  5*1=5  6*1=6  7*1=7  8*1=8  9*1=9
1*2=2  2*2=4  3*2=6  4*2=8  5*2=10 6*2=12 7*2=14 8*2=16 9*2=18
1*3=3  2*3=6  3*3=9  4*3=12 5*3=15 6*3=18 7*3=21 8*3=24 9*3=27
1*4=4  2*4=8  3*4=12 4*4=16 5*4=20 6*4=24 7*4=28 8*4=32 9*4=36
1*5=5  2*5=10 3*5=15 4*5=20 5*5=25 6*5=30 7*5=35 8*5=40 9*5=45
1*6=6  2*6=12 3*6=18 4*6=24 5*6=30 6*6=36 7*6=42 8*6=48 9*6=54
1*7=7  2*7=14 3*7=21 4*7=28 5*7=35 6*7=42 7*7=49 8*7=56 9*7=63
1*8=8  2*8=16 3*8=24 4*8=32 5*8=40 6*8=48 7*8=56 8*8=64 9*8=72
1*9=9  2*9=18 3*9=27 4*9=36 5*9=45 6*9=54 7*9=63 8*9=72 9*9=81
```

重複結構

- while敘述
  - while敘述流程圖：



重複結構

- 語法：

```
資料型態 變數名稱=初值;
while (判斷條件 )
{
        程式敘述區;
        增量值;
}
```

- 說明：
  - 資料型態：
    - 計數器的資料型態。
  - 變數名稱：
    - 計數器的變數名稱。

重複結構

- 初值：
  - 計數器的起始值。
- 判斷條件：
  - 當while迴圈的判斷條件結果為false時，就會結束迴圈，並跳離while迴圈。
- 增量值：
  - 增加（或減少）計數器的值（即 改變變數的值）。

執行步驟：

1.先判斷是否要執行while迴圈；
   - 如果條件運算式為真，則執行while迴圈內的敘述；
   - 如果條件運算式為假，則不執行while迴圈內的敘述。

2.於while迴圈內，須有增加或減少控制變數的值，可視使用者的需求來作控制，再重複步驟1。

重複結構

例如：

```
int i=0;
while(i <=5)
{
    ⋮
   程式敘述區;
    ⋮
   i++;
}
```

重複結構

- 程式：

```
public class CH03_09
{
    public static void main(String[] args)
    {
        int n = 1, sum = 0;

        while (n <= 10)
        {
            System.out.print("n=" + n);
            sum += n;
            System.out.println("\t累加值=" + sum);
            n++;
        }
        System.out.println("迴圈結束");
    }
}
```

重複結構

```
1  public class CH03_09
2  {
3     public static void main(String[] args)
4     {
5        int n=1,sum=0;
6
7        while(n<=10)
8        {
9           System.out.print("n=" + n);
10          sum+=n;
11          System.out.println("\t累加值=" + sum);
12          n++;
13       }
14       System.out.println("迴圈結束");
15    }
16 }
```

重複結構

- 執行結果：

```
n=1  累加值=1
n=2  累加值=3
n=3  累加值=6
n=4  累加值=10
n=5  累加值=15
n=6  累加值=21
n=7  累加值=28
n=8  累加值=36
n=9  累加值=45
n=10     累加值=55
迴圈結束
```

- 說明：
  - 行05：
    - 設定變數，並給予初值。

重複結構

- 行07~行13：
  - 螢幕輸出。
  - 計算累加的結果。
  - 行07：
  - 設定while迴圈。
    - 包含設定迴圈的判斷條件（即 n <= 10 ）。
    - 因為控制變數n從1開始，一直執行到n=11時，才會因為不符合判斷條件而離開，故此迴圈（行07~行13）共要執行10次，且迴圈執行完畢時，n會等於11。
  - 行09：
    - 螢幕輸出。
  - 行10：
    - 計算總和。

重複結構

- 行11：
  - 螢幕輸出。

    螢幕輸出時，先讓游標移到下一個水平定位（\t），再輸出「累加值=」及sum的值。
- 行12：
  - 控制變數加1。

    將控制變數n加1，再回到程式的第7行，檢查是否符合條件運算式，如果運算式的boolean值為false，會跳到第14行，再繼續執行程式。

- 行14：
  - 螢幕輸出。

重複結構

- 課堂練習：
  - 請將此（CH03_09）程式，用『for迴圈』重新改寫。

重複結構

- do-while敘述
  - 語法：

```
資料型態 變數名稱=初值;
do
{
        程式敘述區;
        遞增量;
} while (判斷條件 );
```

  - 說明：
    - 資料型態：
      - 計數器的資料型態。

重複結構

- 變數名稱：
  - 計數器的變數名稱。
- 初值：
  - 計數器的起始值。
- 判斷條件：
  - 當while迴圈的判斷條件結果為false時，就會結束迴圈，並跳離while迴圈。

◆ 執行步驟：

1. 先執行do-while迴圈的敘述一次，再判斷是否要再次執行do-while迴圈；
   - 如果條件運算式為真，則再次執行do-while迴圈內的敘述；如果條件運算式為假，則不再執行do-while迴圈內的敘述。

重複結構

2. 於do-while迴圈內，須有增加或減少控制變數的值，可視使用者的需求來作控制，再重複步驟1。

例如：

```
int i=0;
do
{
    ⋮
   程式敘述區;
    ⋮
   i++;
} while(i <=5);
```

重複結構

注意：

do-while敘述類似while敘述，兩者的差別是條件運算式所在的前後之分。

- while內的敘述，不一定會被執行。
  - 先判斷是否符合判斷條件，

    若符合判斷條件，則執行迴圈中的敘述。

    若不符合判斷條件，則不執行迴圈中的敘述。
- do-while內的敘述，至少會被執行一次。
  - 先執行迴圈中的敘述一次，然後再判斷是否符合判斷條件，

    若符合判斷條件，則再次執行迴圈中的敘述。

    若不符合判斷條件，則不執行迴圈中的敘述。

重複結構

- 程式：

```
public class CH03_10
{
    public static void main(String[] args)
    {
        int n = 40, m = 180;
        int temp = 0;

        System.out.println("n=" + n + "，m=" + m);

        do
        {
            temp = m % n;
            m = n;
            n = temp;
        } while (n != 0);
        System.out.println("兩數的最大公因數為" + m);
    }
}
```

重複結構

```
1  public class CH03_10
2  {
3     public static void main(String[] args)
4     {
5        int n = 40, m = 180;
6        int temp = 0;
7
8        System.out.println("n=" + n + "，m=" + m);
9
10       do
11       {
12          temp = m % n;
13          m = n;
14          n = temp;
15       } while (n != 0);
16       System.out.println("兩數的最大公因數為" + m);
17    }
18 }
```

- 執行結果：

```
n=40，m=180
兩數的最大公因數為20
```

重複結構

- 說明：
  - 行05：
    - 宣告變數（要求最大公因數的兩個數字），並指定初值。
  - 行06：
    - 宣告變數（作為n與m變數交換時的暫存變數），並指定初值。
  - 行08：
    - 螢幕輸出n與m的值。
  - 行10~行15：
    - 以「輾轉相除法」找出最大公因數。
    - 行10：
      - 為do-while迴圈的進入點。

重複結構

- 行12：
  - 將m%n的值（餘數）指定給temp。
  - 此處的m必須大於n。
- 行13 ~ 行14：
  - 利用temp將n與m的值對調，因為此時的n值大於m值。
- 行15：
  - 判斷do-while迴圈是否再次執行。

    如果條件運算式為真，則再次執行do-while迴圈內的敘述。

    如果條件運算式為假，則不再執行do-while迴圈內的敘述。

- 行16：
  - 螢幕輸出。

重複結構

- 無窮迴圈
  - 在設定迴圈控制的條件運算式（即 判斷條件）時，須注意不可使條件運算式的結果恆成立，否則會形成無窮迴圈。
  - 以下列出幾個常見的無窮迴圈的例子，請避免使用：
    - while(true){ }
      - 造成「無窮迴圈」的原因：while敘述永遠為true。
    - for(;;){ }
      - 造成「無窮迴圈」的原因：for敘述沒有設定任何條件。
    - for(int i=1;i>0;i++){ }
      - 造成「無窮迴圈」的原因：判斷條件永遠為true。

分支結構

- break敘述
  - break有兩種用法：
    - 在迴圈中使用break時：
      - 當程式執行到迴圈中的break時，會終止迴圈的執行，並跳出迴圈。
    - 搭配「標記」使用：
      - 可以使程式跳到指定標記（範圍）外的下一行繼續執行。
  - 語法1（在迴圈中使用）：

```
break;
```

    - 說明：
      - 終止迴圈的執行，並跳出迴圈。

分支結構

- 程式：

```
public class CH03_11
{
    public static void main(String[] args)
    {
        int i, num = 1;
        for (i = 0; i < 10; i++)
        {
            num *= 2;
            if (num > 20)
                break;
        }
        System.out.printf("break被執行，此時的 i=%d，num=%d", i, num);
    }
}
```

分支結構

```
1  public class CH03_11
2  {
3     public static void main(String[] args)
4     {
5        int i, num = 1;
6        for (i = 0; i < 10; i++)
7        {
8           num *= 2;
9           if (num > 20)
10             break;
11       }
12       System.out.printf("break被執行，此時的 i=%d，num=%d", i, num);
13    }
14 }
```

- 執行結果：

```
break被執行，此時的 i=4，num=32
```

分支結構

- 說明：
  - 行05：
    - 宣告變數，並指定初值。
  - 行06~行11：
    - 計算2^{10}的結果。
    - 行06：

      設定for迴圈。

      包含設定迴圈的起始值（即 int i = 0）、判斷條件（即 i < 10）及遞增值（即 i++）。

      因為控制變數i從0開始，一直執行到i=10時，才會因為不符合判斷條件而離開；原此迴圈（行06~行11）共要執行11次，且迴圈執行完畢時，i會等於10，然而，此迴圈中又包含了單向選擇（行09~行10），故此迴圈（行06~行11）共會執行5次，且迴圈執行完畢時，i會等於4。

分支結構

- 行08：

  計算num*= 2（即 計算2^n）。

- 行09：

  單向選擇，條件判斷式若條件成立（即 結果為True），則執行下一行的敘述；若條件不成立（即 結果為False），則跳過下一行的敘述。

  註：

  因為只有一行敘述，故可省略大括號。

- 行10：

  若行09的條件成立，則會直接跳離for迴圈，繼續執行迴圈外的第12行。

- 行12：
  - 螢幕輸出。
  - 此處使用「格式化」輸出。

分支結構

- 語法2（搭配「標記」使用）：

```
標記名稱：
{
        ⋮
    break 標記名稱;
        ⋮
}
```

分支結構

- 說明：
  - 程式碼本來是一行一行的敘述，透過標記（Label）的使用，可以把某個範圍的程式碼，全部刮起來成為一個區塊；利用break和標記的搭配使用，便可以讓程式碼從某行程式碼，一口氣跳出整個區塊，到該區塊外的第一行程式繼續執行。
  - 「標記」可自行命名，但要根據識別字的規則，然後在這個名稱後面加上冒號，用來當作是程式區塊的起始記號。
  - 接著利用左、右大括號，將程式區塊範圍標示起來。

分支結構

- 程式：

```
public class CH03_12
{
    public static void main(String[] args)
    {
        int i, j;

        for (i = 1; i < 10; i++)
        {
            for (j = 1; j <= i; j++)
            {
                if (j == 5)
                    break;
                System.out.print(j);
            }
            System.out.println();
        }
        System.out.println();
        System.out.println("跳出雙層迴圈");
```

分支結構

```
        out1:
        {
         for (i = 1; i < 10; i++)
         {
             for (j = 1; j <= i; j++)
             {
                 if (j == 5)
                     break out1;
                 System.out.print(j);
             }
             System.out.println();
         }
         System.out.println();
    }
}
```

分支結構

```
1  public class CH03_12
2  {
3     public static void main(String[] args)
4     {
5        int i, j;
6
7        for (i = 1; i < 10; i++)
8        {
9           for (j = 1; j <= i; j++)
10          {
11             if (j == 5)
12                break;
13             System.out.print(j);
14          }
15          System.out.println();
16       }
17       System.out.println();
18       System.out.println("跳出雙層迴圈");
19
20       out1:
21       {
22          for (i = 1; i < 10; i++)
23          {
24             for (j = 1; j <= i; j++)
25             {
26                if (j == 5)
```

分支結構

```
27                  break out1;
28                System.out.print(j);
29              }
30              System.out.println();
31          }
32          System.out.println();
33      }
34  }
35  }
```

- 執行結果：

```
1
12
123
1234
1234
1234
1234
1234
1234

跳出雙層迴圈
1
12
123
1234
1234
```

分支結構

- 說明：
  - 行05：
    - 宣告變數。
  - 行07~行16：
    - 顯示數字1~9。
    - 行07：

      設定外層for迴圈。

      包含設定迴圈的起始值（即 i = 1）、判斷條件（即 i <= 9）及遞增值（即 i++）。

      因為控制變數i從1開始，一直執行到i=10時，才會因為不符合判斷條件而離開，故此迴圈（行08~行16）共要執行9次，且迴圈執行完畢時，i會等於10。

分支結構

行09：

設定內層for迴圈。

包含設定迴圈的起始值（即 j = 1）、判斷條件（即 j <= i）及遞增值（即 j++）。

因為控制變數j從1開始，一直執行到j<=i時，才會因為不符合判斷條件而離開；然而，此迴圈中又包含了單向選擇（行11~行12），故此迴圈（行09~行14）共會執行5次，且迴圈執行完畢時，j會等於5。

行11：

單向選擇。

行12：

中斷此迴圈執行，跳離此迴圈（行10~行14）。

行13：

螢幕輸出。

分支結構

行15：

螢幕輸出。

- 行17~18：
  - 螢幕輸出。
- 行20~行33：
  - 顯示數字1~9。
  - 行20

    設定標記out1。

    行22：

    設定外層for迴圈。

    包含設定迴圈的起始值（即 i = 1）、判斷條件（即 i <= 9）及遞增值（即 i++）。

    因為控制變數i從1開始，一直執行到i=10時，才會因為不符合判斷條件而離開，故此迴圈（行22~行31）共要執行9次，且迴圈執行完畢時，i會等於10。

分支結構

因為在內層迴圈（行24~行29）中，包含了單向選擇，且在行20又有標記範圍，所以實際上，此巢狀迴圈（行22~行31）只會被執行5次，且迴圈執行完畢時，i會等於5。

行24：

設定內層for迴圈。

包含設定迴圈的起始值（即 j = 1）、判斷條件（即 j <= i）及遞增值（即 j++）。

因為控制變數j從1開始，一直執行到j<=i時，才會因為不符合判斷條件而離開，然而，此迴圈中又包含了單向選擇（行26~行27）且配合標記，故此迴圈（行24~行29）共會執行5次，且迴圈執行完畢時，j會等於5。

行26：

單向選擇。

分支結構

行27：

中斷此迴圈的執行，且跳離此巢狀迴圈（行22~行31）到標記區塊外的第一行程式（行34）繼續執行。

行28：

螢幕輸出。

行30：

螢幕輸出。

行32：

螢幕輸出。

註：

當j==5時，程式會跳到標記區塊外的第一行程式（行34）繼續執行，故不會執行行32的敘述。

分支結構

註：

1、行12的break中斷敘述只會跳出行09~行14的for迴圈（內圈）。

- 行09~行14的執行過程如下：

  i=1→j=1→顯示結果：1

  i=2→j=1~2→顯示結果：1 2

  i=3→j=1~3→顯示結果：1 2 3

  ⋮

  i=5→j=1~5→顯示結果：1 2 3 4（不會顯示5，因為已經跳出迴圈）

  i=6→j=1~6→顯示結果：1 2 3 4

  ⋮

  i=9→j=1~9→顯示結果：1 2 3 4

  i=10（i<10不成立，結束迴圈）

分支結構

2、行20設定中斷標記的名稱（out1），當執行到行27時（break out1;），會跳出整個標記區塊範圍。

- 行22~行31的執行過程如下：

  i=1→j=1→顯示結果：1

  i=2→j=1~2→顯示結果：1 2

  i=3→j=1~3→顯示結果：1 2 3

  ⋮

  i=5→j=1~5→顯示結果：1 2 3 4（不會顯示5，因為已經跳出break中斷標記範圍外的第一行程式（行34）繼續執行，且程式結束）

分支結構

- continue敘述
  - continue有兩種用法：
    - 在迴圈中使用continue時：
      - 功能是強迫for、while、do-while等迴圈敘述，結束正在迴圈本體區塊內進行的程序，而將控制權轉移到迴圈開始處（也就是跳過該迴圈剩下的敘述，重新執行下一次的迴圈）。
    - 搭配「標記」使用：
      - 可以使程式跳到指定標記（範圍）的迴圈繼續執行。
  - 語法1（在迴圈中使用）：

    continue;

    - 說明：
      - 跳過該迴圈剩下的敘述，重新執行下一次的迴圈。

分支結構

- 程式：

```
public class CH03_13
{
    public static void main(String[] args)
    {
        int i, j = 0, num = 1;
        for (i = 0; i < 10; i++)
        {
            num *= 2;
            if (num > 20)
                continue;
            j++;
        }
        System.out.printf("i=%d，j=%d，num=%d", i, j, num);
    }
}
```

分支結構

```
1  public class CH03_13
2  {
3     public static void main(String[] args)
4     {
5        int i, j = 0, num = 1;
6        for (i = 0; i < 10; i++)
7        {
8           num *= 2;
9           if (num > 20)
10             continue;
11          j++;
12       }
13       System.out.printf("i=%d，j=%d，num=%d", i, j, num);
14    }
15 }
```

- 執行結果：

```
i=10，j=4，num=1024
```

分支結構

- 說明：
  - 行05：
    - 宣告變數，並指定初值。
  - 行06~行12：
    - 計算2^{10}。
    - 行06：

      設定for迴圈。

      包含設定迴圈的起始值（即 int i = 0）、判斷條件（即 i < 10）及遞增值（即 i++）。

      因為控制變數i從0開始，一直執行到i=10時，才會因為不符合判斷條件而離開；然而，此迴圈又包含單項選擇（行09~行10），是故此迴圈（行06~行12）共會執行11次，且迴圈執行完畢時，i會等於10。

      行08：

      計算num*=2（即 2^n）。

分支結構

行09：

單向選擇，條件判斷式若條件成立（即 結果為True），則執行下一行的敘述；若條件不成立（即 結果為False），則跳過下一行的敘述。

註：

因為只有一行敘述，故可省略大括號。

行10：

若行09的條件成立，則會結束此次for迴圈的執行，繼續下一次迴圈的執行。

行11：

j加1。

- 行13：
  - 螢幕輸出。
  - 此處使用「格式化」輸出。

分支結構

- 課堂練習：
  - 請將此（CH03_13）程式，重新改寫為『計算任何正整數的任何次方』。

分支結構

- 語法2（搭配「標記」使用）：

```
標記名稱：
    for(;;)（或while、do-while）
    {
        ⋮
      continue 標記名稱;
        ⋮
    }
```

分支結構

- 說明：
  - 利用continue和標記的搭配使用，便可以讓程式碼從某行程式碼，跳回到標記的位置，繼續執行迴圈程式。
  - continue標記的位置，在for、while或do-while迴圈的前面。
  - 「標記」可自行命名，但要根據識別字的規則。

  註：

  因為continue只能用在for、while或do-while迴圈，故可省略大括號（直接以for、while或do-while迴圈為範圍）。

分支結構

- 程式：

```
public class CH03_14
{
    public static void main(String[] args)
    {
        int i, j;

        for (i = 1; i < 10; i++)
        {
            for (j = 1; j <= i; j++)
            {
                if (j == 5)
                    continue;
                System.out.print(j);
            }
            System.out.println();
        }
        System.out.println();
```

分支結構

- 程式：

```
out1:
for (i = 1; i < 10; i++)
{
    for (j = 1; j <= i; j++)
    {
        if (j == 5)
            continue out1;
        System.out.print(j);
    }
    System.out.println();
}
System.out.println();
  }
}
```

分支結構

```
1  public class CH03_14
2  {
3     public static void main(String[] args)
4     {
5        int i, j;
6
7        for (i = 1; i < 10; i++)
8        {
9           for (j = 1; j <= i; j++)
10          {
11             if (j == 5)
12                continue;
13             System.out.print(j);
14          }
15          System.out.println();
16       }
17       System.out.println();
18
19       out1:
20       for (i = 1; i < 10; i++)
21       {
22          for (j = 1; j <= i; j++)
23          {
24             if (j == 5)
25                continue out1;
26             System.out.print(j);
```

分支結構

```
27            }
28            System.out.println();
29        }
30        System.out.println();
31    }
32 }
```

- 執行結果：

```
1
12
123
1234
1234
12346
123467
1234678
12346789

1
12
123
1234
123412341234123412341234
```

分支結構

- 說明：
  - 行05：
    - 宣告變數。
  - 行07~行16：
    - 顯示數字1~9。
    - 行07：

      設定外層for迴圈。

      包含設定迴圈的起始值（即 i = 1）、判斷條件（即 i <10）及遞增值（即 i++）。

      因為控制變數i從1開始，一直執行到i=10時，才會因為不符合判斷條件而離開，故此迴圈（行07~行16）共要執行9次，且迴圈執行完畢時，i會等於10。

      行09：

      設定內層for迴圈。

分支結構

包含設定迴圈的起始值（即 j = 1）、判斷條件（即 j <= i）及遞增值（即 j++）。

因為控制變數j從1開始，一直執行到j<=i時，才會因為不符合判斷條件而離開，但這迴圈中又包含了單向選擇（行11~行12），故此迴圈（行09~行14）共要執行9次，且迴圈執行完畢時，j會等於10。

行11：

單向選擇。

行12：

繼續此迴圈（行09~行14）的執行。

行13：

螢幕輸出。

行15：

螢幕輸出。

分支結構

- 行17：
  - 螢幕輸出。
- 行19~行29：
  - 顯示數字1~9。
  - 行19：

    設定標記out1。
  - 行20~行29：

    顯示數字1~9。

    行20：

    設定外層for迴圈。

    包含設定迴圈的起始值（即 i = 1）、判斷條件（即 i <10）及遞增值（即 i++）。

    因為控制變數i從1開始，一直執行到i=10時，才會因為不符合判斷條件而離開，故此迴圈（行20~行29）共要執行9次，且迴圈執行完畢時，i會等於10。

分支結構

行22：

設定內層for迴圈。

包含設定迴圈的起始值（即 j = 1）、判斷條件（即 j <= i）及遞增值（即 j++）。

因為控制變數j從1開始，一直執行到j<=i時，才會因為不符合判斷條件而離開，然而，此迴圈中又包含了單向選擇（行24~行25）且配合標記，故此迴圈（行22~行27）共要執行5次，且迴圈執行完畢時，j會等於5。

行24：

單向選擇。

行25：

繼續此迴圈（行22~行27）的執行。

行26：

螢幕輸出。

分支結構

行28：

螢幕輸出。

- 行30：

螢幕輸出。

註：

此行是在巢狀for迴圈的範圍外，故會被執行。

分支結構

註：

1. 行12的continue敘述，會讓程式跳過行13，從迴圈的開頭（行9）重新執行下一次的迴圈。
   - 行7~行16的執行過程如下：

     i=1→j=1→顯示結果：1

     i=2→j=1~2→顯示結果：1 2

     i=3→j=1~3→顯示結果：1 2 3

     ⋮

     i=5→j=1~5→顯示結果：1 2 3 4（不會顯示5，因為continue敘述，會回到迴圈的開頭）

     i=6→j=1~6→顯示結果：1 2 3 4 6

     ⋮

     i=9→j=1~9→顯示結果：1 2 3 4 6 7 8 9

     i=10（i<10不成立，結束迴圈）

分支結構

2.行25的continue敘述加上標記，會直接跳過行26的程式，從標記下方的迴圈（行20）重新執行下一次的迴圈。

行20~29的執行過程如下：

i=1→j=1→顯示結果：1

i=2→j=1~2→顯示結果：1 2

i=3→j=1~3→顯示結果：1 2 3

i=4→j=1~4→顯示結果：1 2 3 4

i=5→j=1~5（不會顯示5，因為continue敘述加上標記，會回到標記的下一行，重新執行下一次的迴圈，且因為未執行行28，故直到i=9→j=1~9的顯示結果均未換行；由結果可以看出共有5組1 2 3 4；第一組是i=5的顯示結果；第二組是i=6的顯示結果；第三組是i=7的顯示結果；第四組是i=8的顯示結果；第五組是i=9的顯示結果）

分支結構

- return敘述
  - return敘述可以終止程式目前所在的方法（method）回到呼叫方法的程式敘述。
  - 使用return敘述時，可以將方法中的變數值或運算式值回傳給呼叫的程式敘述，不過回傳值的資料型態要和宣告的資料型態相符合，如果方法不需要回傳值，可以將方法宣告為void資料型態。
  - 語法：

    return 變數或運算式; ←傳回值

    return; ←不傳回值

分支結構

- 程式：

```
public class CH03_15
{
    public static void main(String[] args)
    {
        int ans;
        ans=sum(10);
        System.out.println("1~10的加總");
        System.out.println("ans=" + ans);
    }

    static int sum(int n)
    {
        int sum=0;
        for(int i=1;i<=n;i++)
            sum+=i;
        return sum;
    }
}
```

分支結構

```
1  public class CH03_15
2  {
3     public static void main(String[] args)
4     {
5        int ans;
6        ans=sum(10);
7        System.out.println("1~10的加總");
8        System.out.println("ans=" + ans);
9     }
10
11    static int sum(int n)
12    {
13       int sum=0;
14       for(int i=1;i<=n;i++)
15          sum+=i;
16       return sum;
17    }
18 }
```

- 執行結果：

```
1~10的加總
ans=55
```

分支結構

- 說明：
  - 行05：
    - 宣告變數。
  - 行06：
    - 呼叫（執行）sum方法，並設定變數ans來接收sum方法回傳的值。
    - 10為傳遞的參數。
  - 行07~行08：
    - 螢幕輸出。
  - 行11~行17：
    - sum()方法的定義區。
      - 將方法設定為static，可以直接執行被呼叫和執行，不必透過類別物件。

分支結構

- 行13：

  宣告變數。

- 行14：

  設定for迴圈。

  包含設定迴圈的起始值（即 int i = 1）、判斷條件（即 i <= n）及遞增值（即 i++）。

  因為控制變數i從1開始，一直執行到i>n時，才會因為不符合判斷條件而離開，故此迴圈（行14~行15）共要執行n次，且迴圈執行完畢時，i會等於n+1。

- 行15：

  計算加總。

- 行16：

  將計算結果，傳回至主程式（main）中的變數ans，並繼續執行下一行程式（行07）。

分支結構

- 課堂練習：
  - 請將此（CH03_15）程式，重新改寫為『計算1到輸入數字範圍中，偶數的個數及總和』。

特殊的重複結構

- for-each迴圈
  - for-each可以使迴圈自動化，不用自行動手設定迴圈的計數值、起始值和結束條件值，也不用指定「陣列索引」，好處是避免索引值超過邊界造成錯誤。
  - 語法：

```
for(陣列資料型別 變數名稱：群集名稱)
{
    程式敘述區;
}
```

特殊的重複結構

- 說明：
  - 陣列資料型別：
    - 陣列的資料型別。
  - 變數名稱：
    - 從陣列或集合中讀出的元素值，暫時存放的地方。
  - 群集名稱：
    - 欲讀出元素值的陣列或集合的名稱。

  注意：

  1.迴圈執行的次數，是由陣列元素或集合個數來決定。

  2.若「程式敘述區」中，只有一行程式，大括號可以省略。

特殊的重複結構

- 執行步驟：

  1.設定陣列資料型別、變數名稱與群集名稱。

  2.for-each會自動計算陣列元素或集合的個數，來決定要執行for-each迴圈的次數。

  例如：

```
for ( int i : A)
    System.out.println(i);
```

特殊的重複結構

- 比較for-each迴圈與for迴圈讀取上的不同處。

特殊的重複結構

- 程式（一維陣列）：

```
public class CH03_16
{
    public static void main(String[] ages)
    {
        int A[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
        char B[] = { 'H', 'a', 'p', 'p', 'y' };

        System.out.println("數字陣列");
        for (int i = 0; i < A.length; i++)
            System.out.print(A[i] + " ");
        System.out.println('\n');

        System.out.println("字元陣列");
        for (int i = 0; i < B.length; i++)
            System.out.print(B[i] + " ");
        System.out.println('\n');

        System.out.println("數字陣列");
```

特殊的重複結構

```
        for (int i : A)
            System.out.print(i + " ");
        System.out.println('\n');

        System.out.println("字元陣列");
        for (char i : B)
            System.out.print(i + " ");
        System.out.println('\n');
    }
}
```

特殊的重複結構

```
1  public class CH03_16
2  {
3     public static void main(String[] ages)
4     {
5        int A[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
6        char B[] = { 'H', 'a', 'p', 'p', 'y' };
7
8        System.out.println("數字陣列");
9        for (int i = 0; i < A.length; i++)
10          System.out.print(A[i] + " ");
11       System.out.println('\n');
12
13       System.out.println("字元陣列");
14       for (int i = 0; i < B.length; i++)
15          System.out.print(B[i] + " ");
16       System.out.println('\n');
17
18       System.out.println("數字陣列");
19       for (int i : A)
20          System.out.print(i + " ");
21       System.out.println('\n');
22
23       System.out.println("字元陣列");
24       for (char i : B)
25          System.out.print(i + " ");
26       System.out.println('\n');
```

特殊的重複結構

```
27    }
28 }
```

- 執行結果：

```
數字陣列
123456789

字元陣列
Happy

數字陣列
123456789

字元陣列
Happy
```

- 說明：
  - 行05 ~行06：
    - 宣告陣列，並指定陣列元素的值。

特殊的重複結構

- 行08：
  - 螢幕輸出。
- 行09：
  - 設定for迴圈。
    - 包含設定迴圈的起始值（即 int i = 0）、判斷條件（即 i < A.length；A.length是指陣列的長度，也就是陣列中元素的個數）及遞增值（即 i++）。
    - 因為控制變數i從0開始，一直執行到i = A.length時，才會因為不符合判斷條件而離開（陣列的個數為9，但索引值是從0開始，所以i = A.length時，即表示陣列已結束），故此迴圈（行08~行09）共要執行9次，且迴圈執行完畢時，i會等於9。
- 行10：
  - 螢幕輸出陣列的元素值。

特殊的重複結構

- 行11：
  - 螢幕輸出。
    - 此處為換行（共換2行）。
- 行13：
  - 螢幕輸出。
- 行14：
  - 設定for迴圈。
    - 包含設定迴圈的起始值（即 int i = 0）、判斷條件（即 i < B.length）及遞增值（即 i++）。
    - 因為控制變數i從0開始，一直執行到i = B.length時，才會因為不符合判斷條件而離開，故此迴圈（行12~行13）共要執行5次，且迴圈執行完畢時，i會等於5。
- 行15：
  - 螢幕輸出陣列的元素值。

特殊的重複結構

- 行16：
  - 螢幕輸出。
    - 此處為換行（共換2行）。
- 行18：
  - 螢幕輸出。
- 行19~行20：
  - 利用for-each迴圈，將A陣列中的元素，依序先寫入變數i中，再由行18依序將變數i的內容輸出到螢幕。
- 行21：
  - 螢幕輸出。
    - 此處為換行（共換2行）。
- 行23：
  - 螢幕輸出。

特殊的重複結構

- 行24~行25：
  - 利用for-each迴圈，將B陣列中的元素，依序先寫入變數i中，再由行22依序將變數i的內容輸出到螢幕。
- 行26：
  - 螢幕輸出。
    - 此處為換行（共換2行）。

特殊的重複結構

- 程式（二維陣列）：

```
public class CH03_17
{
    public static void main(String[] ages)
    {
        int A[][] = new int[2][3];

        for (int i = 0; i < 2; i++)
            for (int j = 0; j < 3; j++)
            {
                A[i][j] = i + j;
                System.out.print(A[i][j] + " ");
            }
        System.out.println('\n');

        for (int i[] : A)
            for (int j : i)
                System.out.print(j + " ");
        System.out.println('\n');
```

特殊的重複結構

```
    }
}
```

特殊的重複結構

```
1  public class CH03_17
2  {
3     public static void main(String[] ages)
4     {
5        int A[][] = new int[2][3];
6
7        for (int i = 0; i < 2; i++)
8           for (int j = 0; j < 3; j++)
9           {
10             A[i][j] = i + j;
11             System.out.print(A[i][j] + " ");
12          }
13       System.out.println('\n');
14
15       for (int i[] : A)
16          for (int j : i)
17             System.out.print(j + " ");
18       System.out.println('\n');
19    }
20 }
```

- 執行結果：

```
0 1 2 1 2 3

0 1 2 1 2 3
```

特殊的重複結構

- 說明：
  - 行05：
    - 宣告一個2*3的二維陣列A。
  - 行07~行12：
    - 利用巢狀for迴圈，將元素填入二維陣列A中。
    - 行07
      - 設定外層for迴圈。

        包含設定迴圈的起始值（即 int i = 0）、判斷條件（即 i < 2）及遞增值（即 i++）。

        因為控制變數i從0開始，一直執行到i = 2時，才會因為不符合判斷條件而離開，故此迴圈（行06~行11）共要執行2次，且迴圈執行完畢時，i會等於2。

特殊的重複結構

- 行08：
  - 設定內層for迴圈。

    包含設定迴圈的起始值（即 int j = 0）、判斷條件（即 j < 3）及遞增值（即 j++）。

    因為控制變數j從0開始，一直執行到i = 3時，才會因為不符合判斷條件而離開，故此迴圈（行07~行11）共要執行3次，且迴圈執行完畢時，j會等於3。
- 行10：
  - 計算i+j，並將結果存入A[i][j]陣列中。
- 行11：
  - 螢幕輸出A[i][j]陣列中的元素。
- 行13：
  - 螢幕輸出。

    此處為換行（共換2行）。

特殊的重複結構

- 行15：
  - 外層的for-each迴圈，是表示將是一整組的一維陣列寫入到i[ ]中。
- 行16：
  - 內層的for-each迴圈，則是再針對外層所指定的一維陣列i[ ]的元素值，依序存入j變數中。
- 行17：
  - 依序將變數j的內容輸出到螢幕。
- 行18：
  - 螢幕輸出。

    此處為換行（共換2行）。

綜合練習（1）

題目：**使用選擇結構進行考績評比**

要求：

輸入的考績範圍為0~100。

提示：

1、須將輸入範圍限制在0~100間。

2、等第分為甲（100分）、乙（99~80分）、丙（79~60分）、丁（59~0分）。

綜合練習（2）

題目：**閏年的判斷與應用**

要求：

輸入西元年份（最多4位數字），並利用選擇結構來判斷輸入的西元年份是否為潤年。

提示：

1、須判斷是否輸入超過4位數字。

2、潤年計算的規則是：

四年一潤，百年不潤，四百年一潤。

綜合練習（3）

題目：**計算輸入數字範圍的總和**

要求：

1、輸入兩個數字，並計算輸入數字範圍（含輸入的數字）間數字的總和。

2、輸入的第二個數字，不可以比第一個數字小。

3、請分別用三種迴圈（for、while、do…while）及方法（method）來完成。

資料來源

- 蔡文龍、何嘉益、張志成、張力元，JAVA SE 10基礎必修課，台北市，碁峰資訊股份有限公司，2018年7月，出版。
- 吳燦銘、胡昭民，圖解資料結構-使用Java(第三版)，新北市，博碩文化股份有限公司，2018年5月，出版。
- Ivor Horton，Java 8 教學手冊，台北市，碁峰資訊股份有限公司，2016年9月，出版。
- 李春雄，程式邏輯訓練入門與運用---使用JAVA SE 8，台北市，上奇科技股份有限公司，2016年6月，初版。
- 位元文化，Java 8視窗程式設計，台北市，松崗資產管理股份有限公司，2015年12月，出版。
- Benjamin J Evans、David Flanagan，Java 技術手冊 第六版，台北市，碁峰資訊股份有限公司，2015年7月，出版。
- 蔡文龍、張志成，JAVA SE 8 基礎必修課，台北市，碁峰資訊股份有限公司，2014年11月，出版。
- 陳德來，Java SE 8程式設計實例，台北市，上奇科技股份有限公司，2014年11月，初版。
- 林信良，Java SE 8 技術手冊 ，台北市，碁峰資訊股份有限公司，2014年6月，出版。
- 何嘉益、黃世陽、李篤易、張世杰、黃鳳梅，徐政棠譯，JAVA2 程式設計從零開始--適用JDK7，台北市，上奇資訊股份有限公司，2012年5月，出版。