

流程控制

TYIC桃高資訊社

流程控制(Flow Control)

流程控制，顧名思義，就是控制程式執行的流程、順序
在 **Java** 中，能控制流程的有：

if...else、**switch**、**for**、**while**、**continue**、**break**

這些是**控制流程陳述式(control flow statement)**

與表達陳述式和宣告陳述式不同的是，流程控制陳述式不用分號

if...else

if...else 是用來處理在特定情況下才執行的程式碼

```
if (boolean(條件)) {  
    陳述式...  
} else {  
    陳述式...  
}
```

java

在 **Java** 中，一對大括號表示一個區塊(**block**)

當條件為真時才會執行 **if** 後方區塊的程式碼

否則就會執行 **else** 後方區塊裡的程式碼

else 部分可以省略，省略時如果條件不為真就不會做任何事

if...else


若 **if**、**else** 後方的區塊內只有一行陳述式
則可以不寫區塊，直接寫陳述式

所以可以撰寫下方這種 **if...else if...else** 的語法

```
if (boolean(條件)) {  
    陳述式...  
} else if (boolean(條件)) {  
    陳述式...  
} else {  
    陳述式...  
}
```

java

但除非是 **if...else if...else** 的語法
否則強烈建議使用區塊，避免閱讀錯誤

CVE-2014-1266 

if...else

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input = scanner.next();
        if (input.equals("回憶過去")) {
            System.out.println("痛苦的相思忘不了");
        } else if (input.equals("想念你的笑")) {
            System.out.println("想念你的外套");
        } else if (input.equals("怎麼忍心怪你犯了錯") || input.equals("怎麼忍心讓你受折磨")) {
            System.out.println("是我給你自由過了火");
        } else if (input.equals("我願變成童話裡你愛的那個天使")) {
            System.out.println("張開雙手變成翅膀守護你");
        } else {
            System.out.println("我不知道下一句是什麼");
        }
    }
}
```

回憶過去

痛苦的相思忘不了

console

想念你的笑

想念你的外套

console

怎麼忍心怪你犯了錯

是我給你自由過了火

console

怎麼忍心讓你受折磨

是我給你自由過了火

console

我願變成童話裡你愛的那個天使

張開雙手變成翅膀守護你

console

我是真的不能控制我自己

我不知道下一句是什麼

console



java

switch 流程控制陳述式

switch 的作用是根據不同的傳入值做不同的事

```
switch (傳入值) {  
    case 條件值:  
        陳述式...  
    case 條件值:  
        陳述式...  
    default:  
        陳述式...  
}
```


java

其中 **case** 可以有若干個，而 **default** 可以省略
當傳入值和某個條件值相等時，便會從相等的條件值那行開始
往下執行直到 **switch** 結束，而不管中間的條件值是否相等

switch 流程控制陳述式

在 `switch` 中
可以使用 `break`
讓 `switch` 立刻結束
從而避免 `switch`
一直往下執行的情況

與上個程式碼同樣的功能
在這種情況下寫成 `switch`
比剛剛的 `if...else`
更容易閱讀



```
import java.util.Scanner;

public class Main1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input = scanner.next();
        switch (input) {
            case "回憶過去":
                System.out.println("痛苦的相思忘不了");
                break;
            case "想念你的笑":
                System.out.println("想念你的外套");
                break;
            case "怎麼忍心怪你犯了錯":
            case "怎麼忍心讓你受折磨":
                System.out.println("是我給你自由過了火");
                break;
            case "我願變成童話裡你愛的那個天使":
                System.out.println("張開雙手變成翅膀守護你");
                break;
            default:
                System.out.println("我不知道下一句是什麼");
        }
    }
}
```

java

switch 流程控制陳述式

在 Java 14 中，**switch** 支援了一個 **case** 多個條件值以及可以使用**箭頭** **"->"** 來替代冒號，但兩者不可混用。若使用箭頭，**switch** 只會執行相等條件值箭頭後方的區塊或陳述式，而不會像是用冒號時會一直往下執行，也就不需要 **break**。

```
import java.util.Scanner;

public class Main2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input = scanner.next();
        switch (input) {
            case "回憶過去" -> System.out.println("痛苦的相思忘不了");
            case "想念你的笑" -> System.out.println("想念你的外套");
            case "怎麼忍心怪你犯了錯", "怎麼忍心讓你受折磨" -> System.out.println("是我給你自由過了火");
            case "我願變成童話裡你愛的那個天使" -> System.out.println("張開雙手變成翅膀守護你");
            default -> System.out.println("我不知道下一句是什麼");
        }
    }
}
```

怎麼忍心讓你受折磨
是我給你自由過了火

console



java

switch 表達式

Java 14 加入了 `switch` 表達式，格式與 `switch` 陳述式幾乎相同
但一定需要 `default`，且 `"->"` 後方是要回傳的值，並且回傳值後須加分號
而且若使用冒號或是區塊，需使用 `yield` 來回傳值，並且會終止 `switch`

```
import java.util.Scanner;

public class Main3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String input = scanner.nextLine();
        String output = switch (input) {
            case "回憶過去" -> "痛苦的相思忘不了";
            case "想念你的笑" -> "想念你的外套";
            case "怎麼忍心怪你犯了錯", "怎麼忍心讓你受折磨" -> "是我給你自由過了火";
            case "我願變成童話裡你愛的那個天使" -> "張開雙手變成翅膀守護你";
            default -> {
                if (input.isEmpty()) yield "沒有輸入任何東西";
                yield "我不知道下一句是什麼";
            }
        };
        System.out.println(output);
    }
}
```



現在是星期五晚上
我不知道下一句是什麼 console

沒有輸入任何東西 console

java

三元運算、if...else、switch

三元運算、if...else、switch 有許多相似之處，但仍有不同：

名稱	三元運算	if...else	switch
類型	表達式	陳述式	陳述式 或 表達式
功能	依條件傳回結果	依條件執行陳述式	依傳入值執行陳述式 或傳回結果

switch 作為陳述式時可以與 if...else 互換
而作為表達式時可以與三元運算互換

for

for 是用來重複執行某些程式碼

```
for (①初始化變數; ②boolean(執行條件); ③修改變數) {  
    ④陳述式...  
}
```

java

初始化變數、執行條件、修改變數皆可省略，執行條件預設為真

初始化變數可以是宣告或是賦值，也可以是其他表達式

執行條件為真時才會繼續執行 **for** 迴圈，否則跳出 **for** 迴圈

修改變數可以是賦值，或是其他表達式

執行順序：①->②->③->④->②->③->④->②->③->...

若 **for** 後方的區塊內只有一行陳述式

則可以不寫區塊，直接寫陳述式

for

```
public class Main {  
    public static void main(String[] args) {  
        int i = 1;  
        final String PREFIX = "喜歡你的第";  
        final String SUFFIX = new String("年，我還是沒告白");  
        System.out.println(PREFIX + i++ + "年，我還沒有告白");  
        System.out.println(PREFIX + i++ + SUFFIX);  
        System.out.println(PREFIX + i++ + SUFFIX);  
        System.out.println(PREFIX + i++ + SUFFIX);  
        System.out.println(PREFIX + i++ + "年，我終於告白了");  
    }  
}
```

程式碼有許多行重複
適合使用 **for** 迴圈

java

喜歡你的第1年，我還沒有告白
喜歡你的第2年，我還是沒告白
喜歡你的第3年，我還是沒告白
喜歡你的第4年，我還是沒告白
喜歡你的第5年，我還是沒告白
喜歡你的第6年，我終於告白了

output

```
public class Main {  
    public static void main(String[] args) {  
        for (int i = 1; i <= 6; i++)  
            System.out.printf("喜歡你的第%d年，%s\n", i, switch (i) {  
                case 1 -> "我還沒有告白";  
                case 6 -> "我終於告白了";  
                default -> "我還是沒告白";  
            });  
    }  
}
```



java