

# Chapter6-條件性敘述

## if 敘述

- 語法


```
if (expression){  
    statement1 ;  
    statement2 ;  
    ....  
}
```

- 當小括號內的運算式 (expression) 為 true 時, 才會執行大括號內的敘述 (statement), 否則則跳過大括號內的敘述
- 當大括號內的敘述只有一個時可以省略大括號

## if - else 敘述

- 語法

```
if (expression){  
    statement1 ;  
    statement2 ;  
    ....  
}else{  
    statement1 ;  
    statement2 ;  
    ....  
}
```



if 小括號內條件判斷為 true 時執行

if 小括號內條件判斷為 false 時執行

- 同 if 敘述, else 敘述的左右大括號內只有一個敘述  
時左右大括號可以省略

## if - else if - else 敘述

- 語法

```
if (expression){  
    /* do something */  
}  
else if(expression){  
    /* do something */  
}  
else{  
    /* do something */  
}
```

if 小括號內條件判斷為 true 時  
執行, 為 false 時接者判斷後  
續 else if 的條件(如果有)

else if 小括號內條件判斷為 true  
時執行, 為 false 時接者判斷後續  
else if 的條件(如果有)

先前的 if 以及 else if 條件都沒  
有通過, 則執行 else 部份

- 同 if 以及 else 敘述, else if 敘述的左右大括號內只有一個敘述時左右大括號可以省略

# 巢狀 if 敘述

- 語法

```
if ( expression1 ){  
    /* do something */  
    if ( expression2 ){  
        /* do something */  
    }  
    /* do something */  
}
```

if 小括號內條件判斷 expression 1 為 true 時執行，  
為 false 時接者判斷後續大括號外的 else if 條件(如  
果有)或執行 else(如果沒else if)

要執行內部的 if 內容必須先通過外面 if 的條  
件，且 expression2 的條件也要為 true 才能  
執行內部 if 條件的內容

## if 與 else 配對問題

```
int num;
```

```
scanf("%d", &num);
```

```
if(num >= 0){
```

```
    if(num <= 10)
```

```
        printf("數字介在0到10之間\n");
```

```
}
```

```
else
```

```
    printf("testing\n");
```

**請問此處的 else 是要跟哪個 if 做搭配？**

**Ans: 藍色**

## if 與 else 配對問題 - Dangling-else Ambiguity

```
int num;
```

```
scanf("%d", &num);
```

```
if(num >= 0)
```

```
    if(num <= 10)
```

```
        printf("數字介在0到10之間\n");
```

```
    else
```

```
        printf("testing\n");
```

**請問此處的 else 是要跟哪個 if 做搭配？**

Ans: 紅色，因為此處並沒有大括號做出 scope，因此 else 會與同一層且最近的 if 做搭配

## 避免配對錯 if - else 的方法

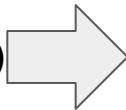
- 縮排在 C 語言並不會影響執行的結果，但可利於閱讀出此處的 if 位在那一層
- 使用大括號明確表示出 if 的 scope (範圍)



# 三元運算子 aka 條件運算子 (Ternary Operator)

- C 語言中三元運算子只有其中一個, 就是 ? :
- 語法: A ? B : C, 其中 A, B, C 分別為 expression (運算式)
- 語意: A 為判斷式
  - 當 A 為 true (非零) 則執行 B 不執行 C 且整個三元運算元的結果為 B 的結果
  - 當 A 為 false (零) 則執行 C 不執行 B 且整個三元運算元的結果為 C 的結果
- 注意: 使用時請注意 precedence

```
int num1, num2, larger;  
scanf("%d %d", &num1, &num2);  
// num1 > num2 ? (larger = num1) : (larger = num2)  
larger = num1 > num2 ? num1 : num2;
```



```
if (num1 > num2){  
    larger = num1;  
}else{  
    larger = num2;  
}
```

# switch 敘述

語法:

```
switch(expression){  
    case constant1:  
        /* do something */  
        break;  
    case constant2:  
        /*do something */  
        break;  
    default:  
        /* do something */  
}
```

- **constant 只能是字元或整數常數**
- switch 會計算括弧內 expression 的內容, 並且比對計算出來的結果跟那一個 label 相同, 並且執行該 label 底下的 statements **一直到 break 結束**
- 若沒有任一個 label 與 expression 相同, 則會執行 default 底下的內容, 執行完後就會跳出, default 需擺在所有 label 的底下

## goto 敘述 (盡量不要使用)

語法:

label\_name1:

```
/* do something */
```

```
goto label_name1;
```

- label\_name1 可自行取名, label\_name1 並不會執行任何計算, 只是當作一個程式的紀錄點
- 執行 goto 時, 程式便會跳到 label\_name1 的部份繼續執行
- 盡量不要使用 goto 在程式裡, 因為在之後維護以及修 bug 時不容易偵測錯誤。

# 課堂練習題

- zerojudge: c382. 加減乘除