

## 第二週

王子銓, 陳毅軒, 吳尚龍

電機通訊程式設計

February 26, 2024

# Outline

1 DOMjudge

2 迴圈

3 位元運算


- 時間限制與記憶體限制: 代表可以使用的執行時間和記憶體大小。
- 題目內容: 敘述題目內容。
- 輸入格式: 說明題目中所敘述的輸入內容與意義。
- 輸出格式: 說明題目中所敘述的輸出內容與意義。
- 技術規格: 規定輸入格式的範圍，保證輸入的內容一定會在這個範圍內，不用特別判斷輸入是否在技術規格內。
- 範例輸入: 範例輸入是簡單舉例可能的輸入與其對應的答案，**不代表輸入只有這些**，所以範例輸入正確不代表程式碼正確。

以第一週 B 小題舉例。

## ● WRONG-ANSWER

Submission details ×

Problem: **B - two\_num\_cal** Submitted: **23:45** Language: **C** Compilation: **successful**

Result: **WRONG-ANSWER** Testcase runs: 

Compilation output

*There were no compiler errors or warnings.*

Close

代表程式碼沒有編譯錯誤，但是程式碼有撰寫錯誤只有部分答案答對。

# COMPILER-ERROR

以第一週 B 小題舉例。

## ● COMPILER-ERROR

Submission details

Problem: **B - two\_num\_cal** Submitted: **23:46** Language: **C** Compilation: **failed**

Compilation output

```
Compiling failed with exitcode 1, compiler output:
b.c: In function 'main':
b.c:7:28: error: expected ';' before 'scanf'
  7 |     scanf("%d %d ", &a, &b)
    |                        ^
    |                        ;
  8 |     scanf("%c", &c);
    |     ~~~~~~
```

Close

代表程式碼有編譯錯誤，裡面會寫出錯誤的地方。

以第一週 B 小題舉例。

## ● TIMELIMIT

Submission details


Problem: B - two\_num\_cal

Submitted: 00:07

Language: C

Compilation: successful

Result: TIMELIMIT

Testcase runs: 

Compilation output

There were no compiler errors or warnings.

Close

代表程式碼執行時間超過時間限制，有可能是計算太久或是進入無窮迴圈。

# NO-OUTPUT

以第一週 B 小題舉例。

## ● NO-OUTPUT

Submission details

Problem: **B - two\_num\_cal**

Submitted: **00:08**

Language: **C**

Compilation: **successful**

Result: **NO-OUTPUT**

Testcase runs:

Compilation output

*There were no compiler errors or warnings.*

Close

程式碼沒有輸出，可能判斷式跳過了輸出。

# RUN-ERROR

以第一週 B 小題舉例。

## ● RUN-ERROR

Submission details ✕

Problem: B - two\_num\_cal Submitted: 00:10 Language: C Compilation: successful

Result: RUN-ERROR Testcase runs: 0000000000

Compilation output

*There were no compiler errors or warnings.*

Close

程式碼使用了超過限制大小的記憶體。

Submission details ✕

Problem: B - two\_num\_cal Submitted: 00:11 Language: C Compilation: successful

Result: RUN-ERROR Testcase runs: 0000000000

Compilation output

```
b.c: In function 'main':
b.c:15:15: warning: division by zero [-Wdiv-by-zero]
   5 |     int k = 9 / 0;
     |               ^
```

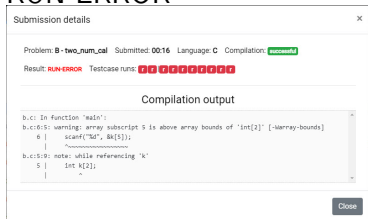
Close

程式碼執行了非法的運算。



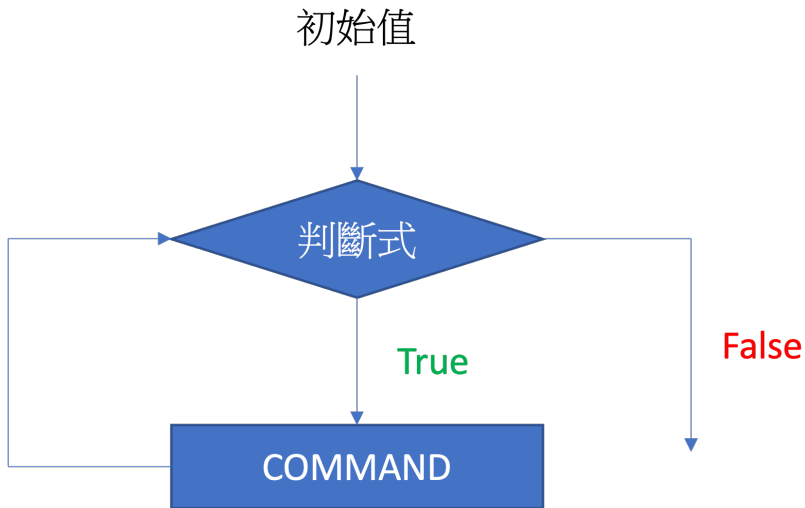
# RUN-ERROR

## ● RUN-ERROR



程式碼使用了未定義的記憶體範圍。

# while 迴圈



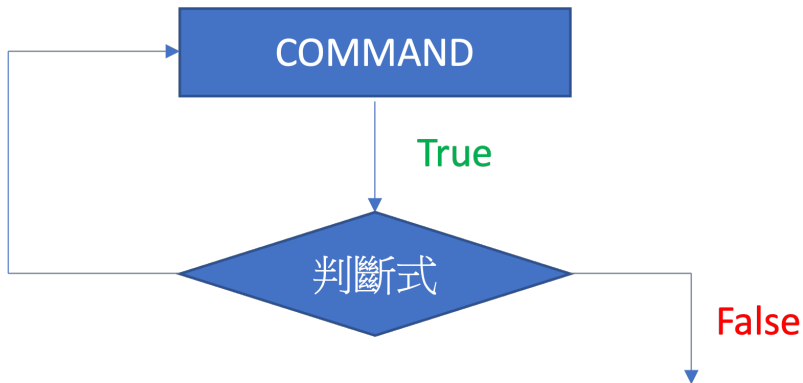
# while 迴圈

```
1 // 語法
2 int i = 1;
3 while (i != 10) {
4     printf("%d ", i);
5     i++;
6 }
```

```
1 // output
2 1 2 3 4 5 6 7 8 9
```

- ① 會依照判斷式裡的結果決定是否執行
- ② 注意迴圈的中止條件，不要形成無窮迴圈

# do while 迴圈



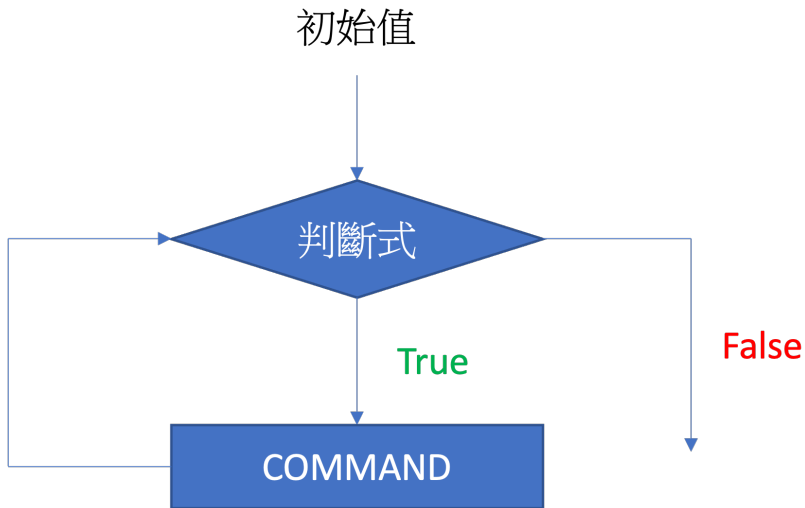
# do while 迴圈

先做動作，若條件仍然達成，重複動作

```
1 // 語法
2 int i = 10;
3 do {
4     printf("%d", i);
5 } while (i != 10);
```

```
1 // output
2 10
```

- 1 就算不符合條件也會執行第一次
- 2 while 最後面記得加分號



# for 迴圈

() 內有初始式、判斷式、調整式，以分號隔開

```
1 for (int i = 0; i < 5; i++) {    // 輸入 5 個數字，並輸出
2     int x; scanf("%d", &x);
3     printf("%d ", x);
4 }
```

```
1 // input
2 1 2 3 4 5
3 // output
4 1 2 3 4 5
```

- 1 初始式: 只會執行一次，然後進入判斷式
- 2 判斷式: 當判斷式為 true，進入 block
- 3 調整式: block 執行完後，進入調整式，然後再進入判斷式
- 4 每個 Statement 中皆可填入不只一個 Expression

想要提前終止迴圈中本次的操作時可使用

```
1  for (int i = 0; i < 10; i++) {  
2      if (i < 5) continue;  
3      printf("%d ", i);  
4  }
```

```
1  // output  
2  5 6 7 8 9
```

在上方區塊第 2 行，若執行了 `continue`，就不會執行第三行的 `printf`



想要提前終止迴圈可使用

```
1  for (int i = 0; i < 10; i++) {  
2      if (i > 5) break;  
3      printf("%d ", i);  
4  }
```

```
1  // output  
2  0 1 2 3 4 5
```

在上方區塊第 2 行，若  $i$  已經到 6，就會直經 `break`，直接終止迴圈，且不會印出 6

Table: 位元運算符號

運算符號	描述
&	AND 運算
	OR 運算
^	XOR 運算
~	NOT 運算
«	左移運算
»	右移運算

# 位元運算 &

Table: AND 運算符的真值表

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

```
1 // 語法
2 int a = 10, b = 9;
3 printf("%d", a & b);
```

```
1 // output
2 // a = 1010
3 // b = 1001
4 1000 -> 8
```

Table: OR 運算符的真值表

A	B	A   B
0	0	0
0	1	1
1	0	1
1	1	1

```
1 // 語法
2 int a = 10, b = 9;
3 printf("%d", a | b);
```

```
1 // output
2 // a = 1010
3 // b = 1001
4 1011 -> 11
```

# 位元運算 ^

Table: XOR 運算符的真值表

A	B	$A \wedge B$
0	0	0
0	1	1
1	0	1
1	1	0

```
1 // 語法
2 int a = 10, b = 9;
3 printf("%d", a ^ b);
```

```
1 // output
2 // a = 1010
3 // b = 1001
4 0011 -> 3
```

# 位元運算 ~

~ 將二進位數字 0,1 反轉，要特別注意是否有號。

A	~A
0	1
1	0

```
1 // 語法
2 int a = 10;
3 printf("%d", ~a);
4 unsigned int b = 10;
5 printf("%d", ~b);
6 printf("%u", ~a); //輸出無號數
```

```
1 // output
2 // %d print signed int
3 // a = 01010
4 10101 -> -11
5 // %u print unsigned int
6 // b = 0000 0000 0000 0000 0000 0000 0000 1010
7 1111 1111 1111 1111 1111 1111 1111 0101 -> 4294967285
```

# 位元運算 <<

## << 向左位移數個位元

```
1 // 語法
2 int a = 10;
3 printf("%d", a << 2);
```

```
1 // output
2 // a = 1010
3 1010 00 -> 40
```

# 位元運算 >>

>> 向右位移數個位元

```
1 // 語法
2 int a = 10;
3 printf("%d", a >> 2);
```

```
1 // output
2 // a = 1010
3 0010 -> 2
```