



# 流程控制 - 2

loop

---

作者: 劉宸均



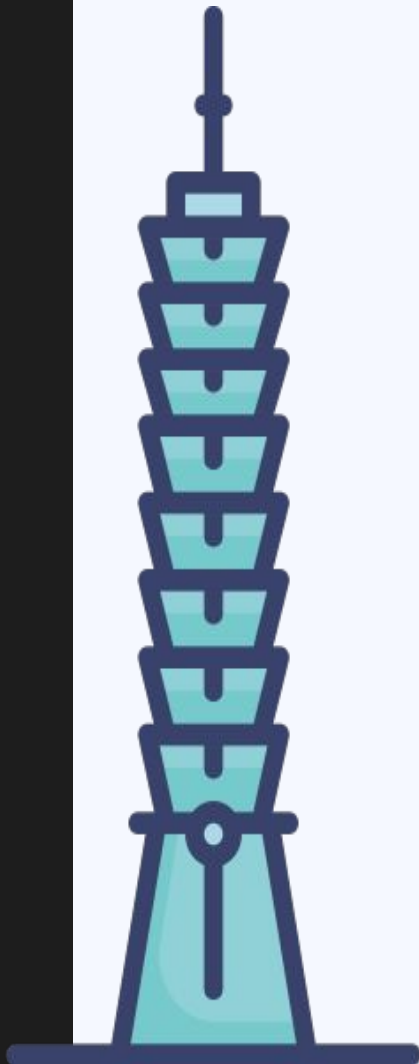
# 迴圈 loop

迴圈可以幹嘛？



## 4. 流程控制

```
1 #include <stdio.h>
2 int main()
3 {
4     printf("hello!\n");
5     printf("hello!\n");
6     printf("hello!\n");
7     printf("hello!\n");
8     printf("hello!\n");
9     printf("hello!\n");
10    printf("hello!\n");
11    printf("hello!\n");
12    printf("hello!\n");
13    printf("hello!\n");
14    printf("hello!\n");
15    printf("hello!\n");
16    printf("hello!\n");
17    printf("hello!\n");
18    printf("hello!\n");
19    printf("hello!\n");
20    printf("hello!\n");
21    printf("hello!\n");
22    printf("hello!\n");
23    printf("hello!\n");
24    printf("hello!\n");
25    printf("hello!\n");
26    printf("hello!\n");
27    printf("hello!\n");
28    printf("hello!\n");
29    printf("hello!\n");
30    printf("hello!\n");
31    printf("hello!\n");
32    printf("hello!\n");
33    printf("hello!\n");
34    printf("hello!\n");
35    printf("hello!\n");
36    printf("hello!\n");
37    printf("hello!\n");
38    printf("hello!\n");
39    printf("hello!\n");
40    printf("hello!\n");
41    printf("hello!\n");
42    printf("hello!\n");
43    printf("hello!\n");
44    printf("hello!\n");
45    printf("hello!\n");
46    printf("hello!\n");
47    printf("hello!\n");
48    printf("hello!\n");
49    printf("hello!\n");
50    printf("hello!\n");
51    printf("hello!\n");
52    printf("hello!\n");
53    printf("hello!\n");
54    printf("hello!\n");
55    printf("hello!\n");
56    printf("hello!\n");
57    printf("hello!\n");
58    printf("hello!\n");
59    printf("hello!\n");
60    printf("hello!\n");
61    printf("hello!\n");
62    printf("hello!\n");
63    printf("hello!\n");
64    printf("hello!\n");
65    printf("hello!\n");
66    printf("hello!\n");
67    printf("hello!\n");
68    printf("hello!\n");
69    printf("hello!\n");
70    printf("hello!\n");
71    printf("hello!\n");
72    printf("hello!\n");
73    printf("hello!\n");
74    printf("hello!\n");
75    printf("hello!\n");
76    printf("hello!\n");
77    printf("hello!\n");
78    printf("hello!\n");
79    printf("hello!\n");
80    printf("hello!\n");
81    printf("hello!\n");
82    printf("hello!\n");
83    printf("hello!\n");
84    printf("hello!\n");
85    printf("hello!\n");
86    printf("hello!\n");
87    printf("hello!\n");
88    printf("hello!\n");
89    printf("hello!\n");
90    printf("hello!\n");
91    printf("hello!\n");
92    printf("hello!\n");
93    printf("hello!\n");
94    printf("hello!\n");
95    printf("hello!\n");
96    printf("hello!\n");
97    printf("hello!\n");
98    printf("hello!\n");
99    printf("hello!\n");
100   printf("hello!\n");
101   printf("hello!\n");
102   }
103 }
```



```
1 #include <stdio.h>
2 int main(){
3     for (int i=0; i<100;i++){
4         printf("Hello!\n");
5     }
6     return 0;
7 }
8
```

# while loop

每次判斷condition是否為真(True),  
若True則執行迴圈內容

```
while (condition) {  
    // 若condition成立則重複執行  
    // ...  
}
```

# while loop

每次判斷condition是否為真(True),  
若True則執行迴圈內容

① True

```
while (condition) {  
    // 若condition成立則重複執行  
    // ...  
}
```

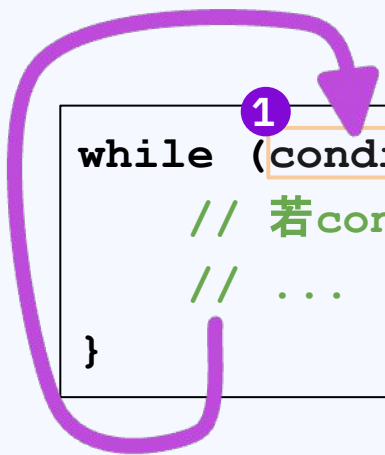
# while loop

每次判斷condition是否為真(True),  
若True則執行迴圈內容

```
while (condition) {  
    2 // 若condition成立則重複執行  
    // ...  
}
```

# while loop

每次判斷condition是否為真(True),  
若True則執行迴圈內容



```
while (condition) {  
    // 若condition成立則重複執行  
    // ...  
}
```

The diagram illustrates the execution of a while loop. A purple arrow starts at the 'condition' part of the 'while' statement, loops around the code block, and points back to the 'while' statement, indicating the repeated nature of the loop. The 'condition' is highlighted with an orange box, and a purple circle with the number '1' is placed next to it. An orange arrow points from the explanatory text to this '1'.

# while loop

每次判斷condition是否為真(True),  
若True則執行迴圈內容

```
① True  
while (condition) {  
    // 若condition成立則重複執行  
    // ...  
}
```



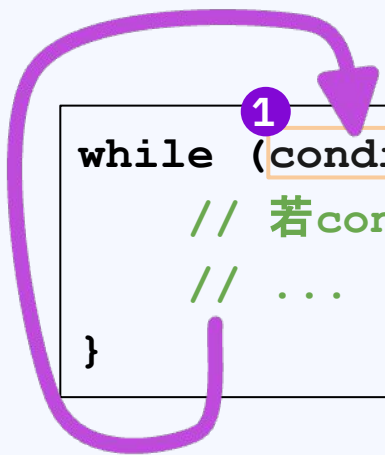
# while loop

每次判斷condition是否為真(True),  
若True則執行迴圈內容

```
while (condition) {  
    2 // 若condition成立則重複執行  
    // ...  
}
```

# while loop

每次判斷condition是否為真(True),  
若True則執行迴圈內容



```
while (condition) {  
    // 若condition成立則重複執行  
    // ...  
}
```

The diagram shows a code block for a while loop. A purple arrow starts at the opening curly brace, loops around the code, and points back to the condition 'condition' in the while statement, illustrating the loop's repetition. A small purple circle with the number '1' is positioned above the condition. An orange line points from the explanatory text to the condition.

# while loop

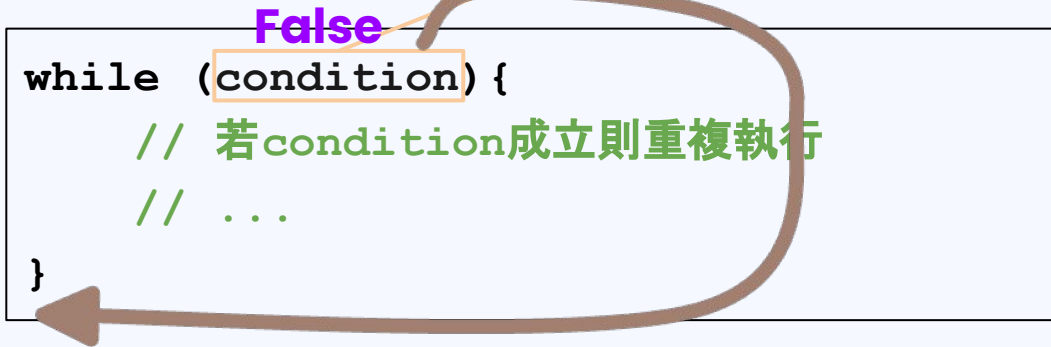
每次判斷condition是否為真(True),  
若True則執行迴圈內容

① False

```
while (condition) {  
    // 若condition成立則重複執行  
    // ...  
}
```

# while loop

每次判斷condition是否為真(True),  
若True則執行迴圈內容



# while loop 範例

loop 迴圈

```
1 #include <stdio.h>
2
3 int main(){
4
5     int i=1;
6     while (i<=10){
7         printf("%d ", i);
8         ++i;
9     }
10    printf("\n");
11
12    return 0;
13 }
```

# while loop 範例

loop 迴圈

```
1 #include <stdio.h>
2
3 int main(){
4
5     int i=1;
6     while (i<=10){
7         printf("%d ", i);
8         ++i;
9     }
10    printf("\n");
11
12    return 0;
13 }
```

1 2 3 4 5 6 7 8 9 10

# 自己玩玩看 while-loop !

試著印 1-100!

# 無窮迴圈

```
while (1){  
    printf("Hello World!\n");  
}
```

可以按Ctrl + C退出



# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

0

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

```
for (1 initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

**① True**

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

```
for (initialization; condition; update) {  
    2 // 若condition成立則重複執行  
    // ...  
}
```

True

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

**① True**

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容



# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

```
for (initialization; condition; update) {  
    2 // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

**1**

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop

碰到for迴圈時先執行此，  
且只會執行一次

for迴圈內執行完先執行此，  
可用於update變數值

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

① False

每次判斷condition是否為真  
(True)，若是則執行迴圈內容

# for loop 範例

控制

loop 迴圈

```
1 #include <stdio.h>
2
3 int main(){
4
5     for (int i=1; i<=10; i++){
6         printf("%d ", i);
7     }
8     printf("\n");
9
10    return 0;
11 }
```

# for loop 範例

```
1 #include <stdio.h>
2
3 int main(){
4
5     for (int i=1; i<=10; i++){
6         printf("%d ", i);
7     }
8     printf("\n");
9
10    return 0;
11 }
```

1 2 3 4 5 6 7 8 9 10

# for, while差異比較

## for loop

碰到for迴圈時先執行此,  
且只會執行一次

for迴圈內執行完先執行此,  
可用於update變數值

```
for (initialization; condition; update) {  
    // 若condition成立則重複執行  
    // ...  
}
```

## while loop

每次判斷condition是否為真  
(True), 若是則執行迴圈內容

```
while (condition) {  
    // 若condition成立則重複執行  
    // ...  
}
```



## 範例：

輸入整數N， 由小到大印出所有  $1 \sim N$  中（包含N）  
的偶數

## 練習—倍數增長

輸入整數N，由小到大印出所有 1~N 中（包含N）7  
的倍數

# do...while loop

```
do{  
    // 首次不判斷 condition  
}while (condition); // 若condition成立則重複執行
```

## while loop

```
while (condition){  
    // 若condition成立則重複執行  
}
```

# do...while loop

```
do{  
    // 首次不判斷 condition  
}while (condition); // 若 condition 成立則重複執行
```

迴圈內部

比較

條件判斷

## while loop

```
while (condition) {  
    // 若 condition 成立則重複執行  
}
```

# 無窮迴圈

```
while (1){  
    printf("Hello World!\n");  
}
```

```
for (int i=0; i==i; i++){  
    printf("%d ", i);  
}
```

可以按Ctrl + C退出

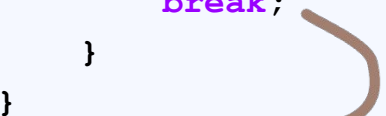
# break/continue

C++ 中用於控制迴圈行為的關鍵字。

## break

**break** 語句在執行時，如果位於 **while**、**for** 或 **switch** 語句中，會立即退出該語句。

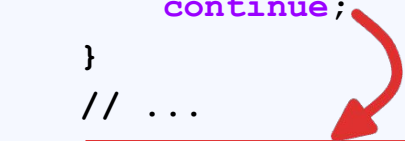
```
for (initialization; condition; update){  
    if (condition to break){  
        break;  
    }  
}
```



## continue

**continue** 語句在執行時，如果位於 **while**、**for** 語句中，會跳過該控制語句中剩餘的語句，並進行下一次的迴圈迭代。

```
for (initialization; condition; update){  
    if (condition to continue){  
        continue;  
    }  
    // ...  
}
```



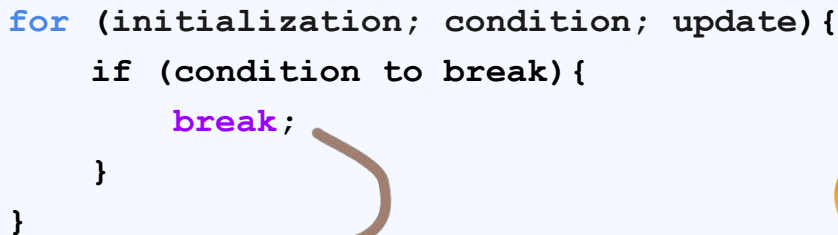
# break/continue

C++ 中用於控制迴圈行為的關鍵字。

## break

**break** 語句在執行時，如果位於 **while**、**for** 或 **switch** 語句中，會立即退出該語句。

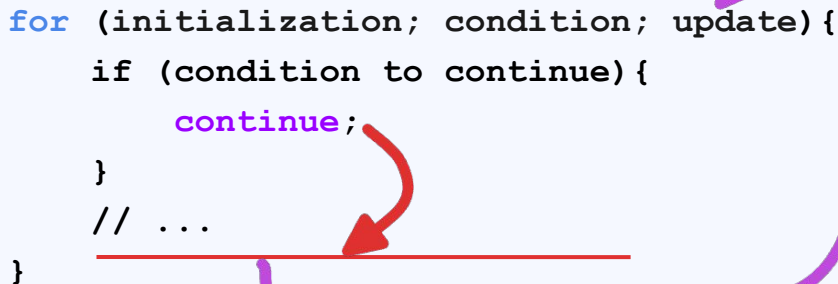
```
for (initialization; condition; update){  
    if (condition to break){  
        break;  
    }  
}
```

A diagram illustrating the effect of the break statement. A brown arrow originates from the 'break;' line within the if block and points to the closing curly brace of the for loop, indicating that the loop is terminated immediately.

## continue

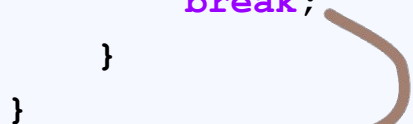
**continue** 語句在執行時，如果位於 **while**、**for** 語句中，會跳過該控制語句中剩餘的語句，並進行下一次的迴圈迭代。

```
for (initialization; condition; update){  
    if (condition to continue){  
        continue;  
    }  
    // ...  
}
```

A diagram illustrating the effect of the continue statement. A red arrow points from the 'continue;' line to the closing curly brace of the if block. A purple arrow then points from the bottom of the if block down to the 'update' part of the for loop, indicating that the rest of the loop body is skipped and the next iteration begins.

# break

```
for (initialization; condition; update){  
    if (condition to break){  
        break;  
    }  
}
```

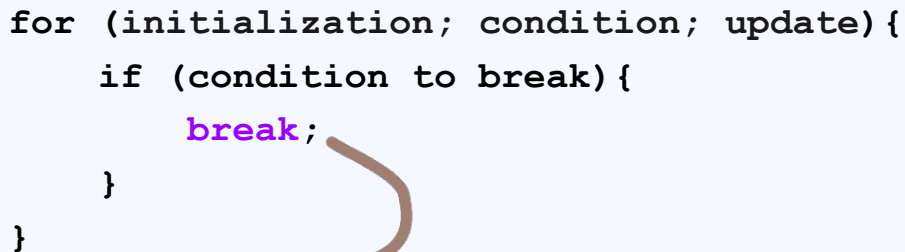


```
1 #include <stdio.h>  
2 int main(){  
3     int x;  
4     for (x=0; x<10; x++){  
5         if (x==5)  
6             break;  
7         printf("%d ", x);  
8     }  
9     printf("\nBreak out of loop at x==%d\n", x);  
10  
11     return 0;  
12 }
```



# break

```
for (initialization; condition; update){  
    if (condition to break){  
        break;  
    }  
}
```

A diagram showing a brown arrow starting from the 'break;' statement and pointing to the closing curly brace of the 'for' loop, indicating that the loop is exited immediately.

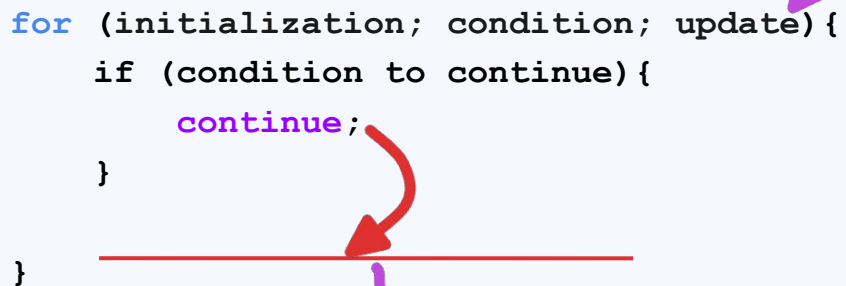
```
1 #include <stdio.h>  
2 int main(){  
3     int x;  
4     for (x=0; x<10; x++){  
5         if (x==5)  
6             break;  
7         printf("%d ", x);  
8     }  
9     printf("\nBreak out of loop at x==%d\n", x);  
10  
11     return 0;  
12 }
```

0 1 2 3 4

Break out of loop at x==5

# continue

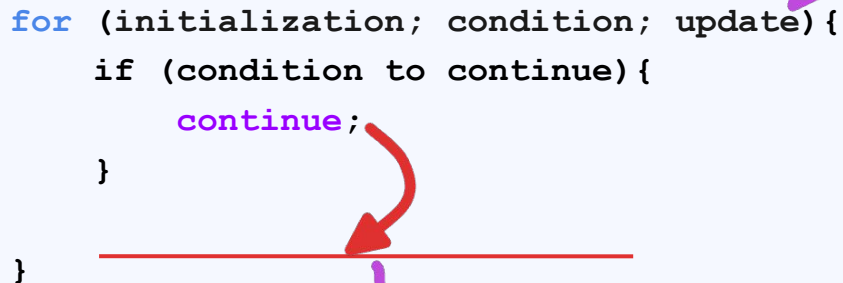
```
for (initialization; condition; update){  
    if (condition to continue){  
        continue;  
    }  
}
```



```
1 #include <stdio.h>  
2 int main(){  
3     int x;  
4     for (x=0; x<10; x++){  
5         if (x==5)  
6             continue;  
7  
8         printf("%d ", x);  
9     }  
10    printf("\n");  
11  
12    return 0;  
13 }
```

# continue

```
for (initialization; condition; update){  
    if (condition to continue){  
        continue;  
    }  
}
```



```
1 #include <stdio.h>  
2 int main(){  
3     int x;  
4     for (x=0; x<10; x++){  
5         if (x==5)  
6             continue;  
7  
8         printf("%d ", x);  
9     }  
10    printf("\n");  
11  
12    return 0;  
13 }
```

0 1 2 3 4 6 7 8 9

# 練習一：猜數字遊戲

遊戲目標：玩家需要在最少次數內猜中電腦隨機選定的 **1 ~ 100** 之間的數字。

猜數字遊戲說明

遊戲名稱：猜數字 (1~100)

遊戲目標

玩家需要在最少次數內猜中電腦隨機選定的 1 到 100 之間的數字。

遊戲規則

電腦隨機生成一個 1 到 100 的數字，並且保密不告訴玩家。

玩家每次輸入一個猜測的數字，電腦會根據玩家的猜測給出提示：

「Too Big」：表示猜測的數字比正確答案大。

「Too Small」：表示猜測的數字比正確答案小。

「BINGO!」：表示玩家的猜測正確。

玩家根據提示繼續猜，直到正確猜出答案為止。

遊戲結束時，電腦會告訴玩家一共猜了多少次。

## 練習二：找出 $N$ 以下的所有質數

說明：輸入一個正整數 $N$ ，印出小於 $N$ 的所有質數

# 巢狀迴圈

```
1  #include <stdio.h>
2  int main(){
3      for (int i=0; i<3; i++){
4          for (int j=0; j<5; j++){
5              printf("i=%d j=%d\n", i, j);
6          }
7      }
8      return 0;
9  }
```

# 巢狀迴圈

```
1  #include <stdio.h>
2  int main(){
3      for (int i=0; i<3; i++){
4          for (int j=0; j<5; j++){
5              printf("i=%d j=%d\n", i, j);
6          }
7      }
8      return 0;
9  }
```

```
i=0 j=0
i=0 j=1
i=0 j=2
i=0 j=3
i=0 j=4
i=1 j=0
i=1 j=1
i=1 j=2
i=1 j=3
i=1 j=4
i=2 j=0
i=2 j=1
i=2 j=2
i=2 j=3
i=2 j=4
```

# 巢狀迴圈範例

輸入正整數N, 印N排的三角形星星:

例:

輸入: 3

輸出:     \*  
          \*\*  
         \*\*\*

輸入: 5

輸出:     \*  
          \*\*  
         \*\*\*  
        \*\*\*\*  
        \*\*\*\*\*



輸入正整數N, 印N排的三角形星星:

```
1  #include <stdio.h>
2  int main(){
3      int N;
4      scanf("%d", &N);
5
6      for (int i=0; i<N; i++){
7          for (int j=0; j<= i; j++){
8              printf("*");
9          }
10         printf("\n");
11     }
12 }
```

# 練習：倒星星

輸入正整數N，代表等腰三角形邊長，  
印N排的倒三角形星星

例： 輸入： 3  
輸出： \*\*\*  
      \*\*  
      \*

輸入： 5  
輸出： \*\*\*\*\*  
      \*\*\*\*\*  
      \*\*\*  
      \*\*  
      \*