計算機程式設計

Computer Programming

Program Control (2)

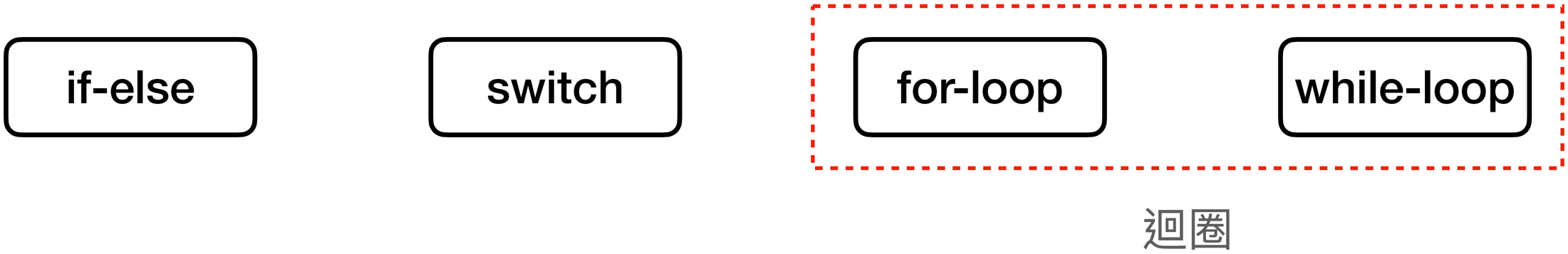
Instructor: 林英嘉

2024/09/30

W4 Slido: #1371156

GitHub repo

Outline: C語言程式流程控制



迴圈

迴圈 (Loop)

- 迴圈是一種讓程式反覆執行某段程式碼的結構，可以減少程式碼的重複性
- 舉例：請計算1到1000的數值總和

```
#include <stdio.h>
int main(){
    int a = 1;
    a += 2;
    a += 3;
    a += 4;
    a += 5;
    ...
}
```

C語言主要迴圈結構

- for
- while
- do-while

for 迴圈 (for - loop) 介紹

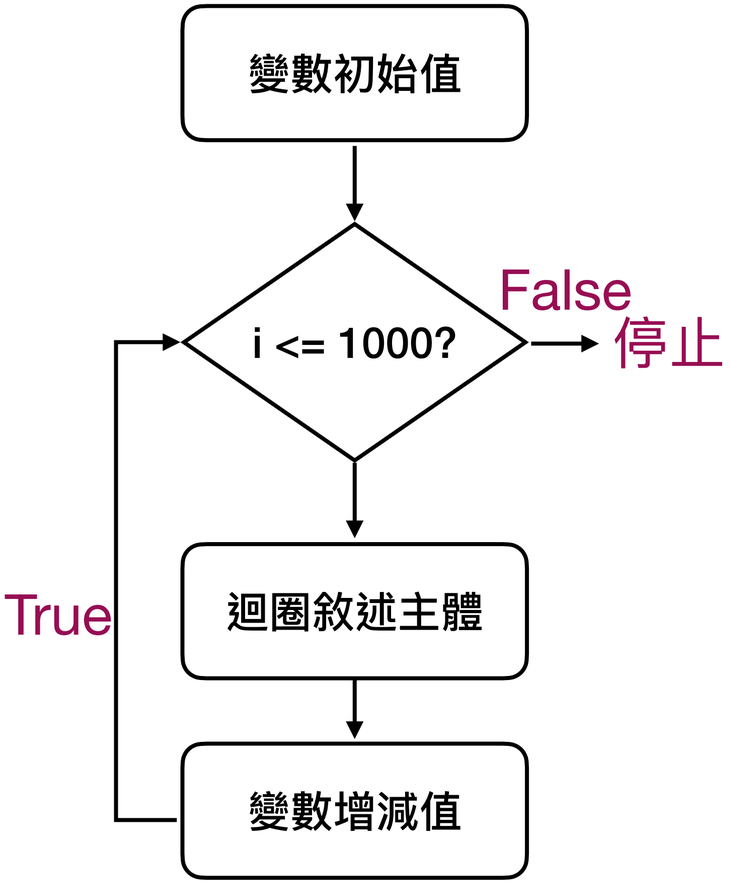
```
for (設定迴圈初值; 迴圈限制; 增減量){
    迴圈敘述主體;
}
```

- 進入迴圈之前先檢查條件 (限制)，條件成立才會執行迴圈敘述主體

for 迴圈 (for - loop) 介紹 vs. 範例

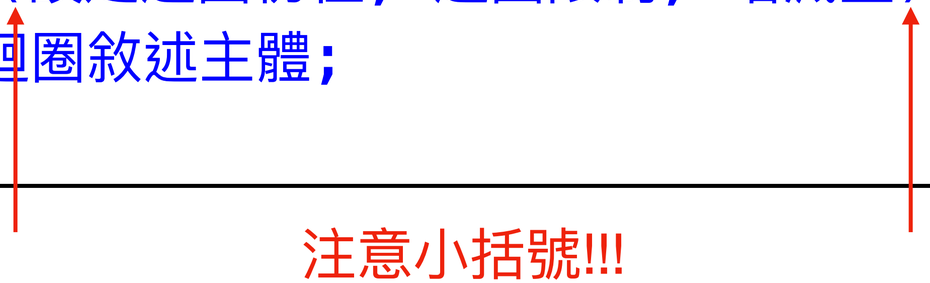
```
for (設定迴圈初值; 迴圈限制; 增減量){
    迴圈敘述主體;
}
```

```
#include <stdio.h>
int main() {
    int sum = 0;
    for (int i = 1; i <= 1000; i++) {
        sum += i;
    }
    printf("1到1000的和是：%d\n", sum);
}
```



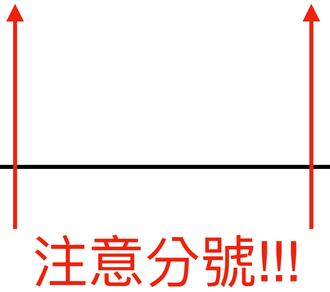
for 迴圈 (for - loop) 介紹：格式結構

```
for (設定迴圈初值; 迴圈限制; 增減量){
    迴圈敘述主體;
}
```



for 迴圈 (for - loop) 介紹：格式結構

```
for (設定迴圈初值; 迴圈限制; 增減量){
    迴圈敘述主體;
}
```



for 迴圈 (for - loop) 介紹：格式結構

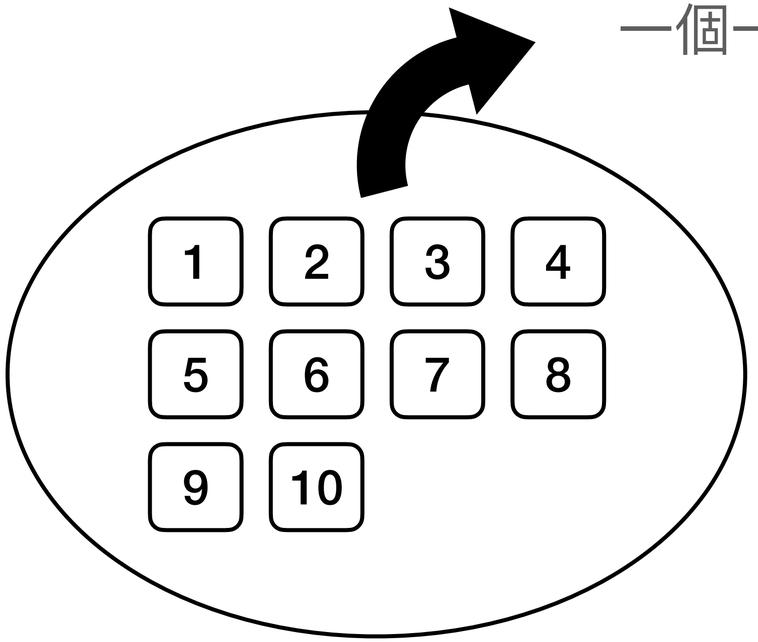
敘述主體開始 (左大括號)

```
for (設定迴圈初值; 迴圈限制; 增減量){
    迴圈敘述主體;
}
```

敘述主體結束 (右大括號)

為什麼for迴圈叫做for迴圈？

- 英文意義：for each




```
for (int i = 1; i <= 10; i++) {
   //迴圈敘述
}
```

for迴圈反向計數

```c
#include <stdio.h>
int main() {
    int i;
    for (i = 10; i >= 1; i--) {
        printf("現在數字為%d\n", i);
    }
}
```

for-loop的邏輯：終點是什麼？

```
for (i = 0; i <= n; i++) ... // i 從 0 到 n
for (i = 0; i < n; i++) ... // i 從 0 到 n - 1
```

```
for (i = n - 1; i >= 0; i--) ... // i 從 n - 1 到 0
for (i = n - 1; i > 0; i--) ... // i 從 n - 1 到 1
```

(小結) 迴圈的條件

- 在C語言中，迴圈需要設定條件
  - 例如：在什麼條件下，開始執行迴圈內的敘述；什麼樣的條件會結束
- 若迴圈的設定不當，則會導致無窮迴圈 (Infinite loops)

無窮迴圈

```c
#include <stdio.h>
int main() {
    int i;
    for (i = 1; ; i++) { // 沒有設終點
        printf("%d\n", i);
    }
}
```

```c
#include <stdio.h>
int main() {
    int i;
    for (i = 1; i+=1 ; i++) { // 終點遠離
        printf("%d\n", i);
    }
}
```

巢狀迴圈：以九九乘法表為例

```c
#include <stdio.h>
int main() {
    int i, j;
    for (i = 1; i <= 9; i++) {
        for (j = 1; j <= 9; j++) {
            printf("%d * %d = %d\n", i, j, i * j);
        }
        printf("\n");
    }
}
```

巢狀迴圈：以三角形為例

```c
#include <stdio.h>
int main() {
    int rows = 5;
    for (int i = 1; i <= rows; i++) {  // 控制行數量
        for (int j = 1; j <= i; j++) {  // 控制每行的星號數量
            printf("* ");
        }
        printf("\n");
    }
}
```

巢狀迴圈：以倒三角形為例

```c
#include <stdio.h>
int main() {
    int rows = 5;
    for (int i = rows; i > 0; i--) {  // 控制行數量
        for (int j = 1; j <= i; j++) {  // 控制每行的星號數量
            printf("* ");
        }
        printf("\n");
    }
}
```

迴圈可不可以提早結束？
break / continue

break 敘述

```
for (設定迴圈初值; 迴圈限制; 增減量){
    迴圈敘述主體;
    break;
}
```

- 在迴圈中加入break，可以使迴圈中斷
- break自己佔一行且需要加上分號

break 敘述

```c
#include <stdio.h>
int main() {
    int i;
    for (i = 1; i+=1 ; i++) { // 終點遠離
        if (i == 10000) {
            break;
        }
        printf("%d\n", i);
    }
}
```

- 注意break的位置，如果想要先印再break的話應該把if段落至於printf之下

continue 敘述

```
for (設定迴圈初值; 迴圈限制; 增減量){
    迴圈敘述主體;
    continue;
}
```

- 在迴圈中加入continue，可以使迴圈直接進入下一次的迴圈
- continue自己佔一行且需要加上分號

continue 敘述

- 只印出奇數，跳過偶數不印

```c
#include <stdio.h>
int main() {
    int i;
    for (i = 1; i <= 10; i++) {
        if (i % 2 == 0) {
            continue;  // 如果 i 是偶數，跳過本次迴圈
        }
        printf("%d\n", i);
    }
}
```

巢狀迴圈下的break行為

```c
#include <stdio.h>
int main() {
    int i, j;
    int stop_num = 6;
    for (i = 1; i <= 9; i++) {
        for (j = 1; j <= 9; j++) {
            printf("%d * %d = %d\n", i, j, i * j);
            if (j == stop_num){
                break;
            }
        }
        printf("\n");
    }
}
```

- break的功能是跳出當層的for-loop，不影響外層

巢狀迴圈下的continue行為

```c
#include <stdio.h>
int main() {
    int i, j;
    for (i = 1; i <= 9; i++) {
        for (j = 1; j <= 9; j++) {
            if ((j == 5) || (j == 7)){
                // 跳過 5 和 7
                continue;
            }
            printf("%d * %d = %d\n", i, j, i * j);
        }
        printf("\n");
    }
}
```

- continue的功能是「跳過」當層的for-loop，不影響外層

C語言主要迴圈結構

- for
- while
- do-while

while 迴圈介紹

```
while (進入迴圈的條件){
    迴圈敘述主體;
}
```

- 進入迴圈之前先檢查條件，條件成立才會執行迴圈敘述主體

Semantic meaning of `while`

- while: 「當...時，... 」
  - While it was raining, I stayed inside.

Semantic meaning of `while`

- while: 「當...時，...」
  - While it was raining, I stayed inside.
    條件　　敘述主體

while 迴圈簡單範例

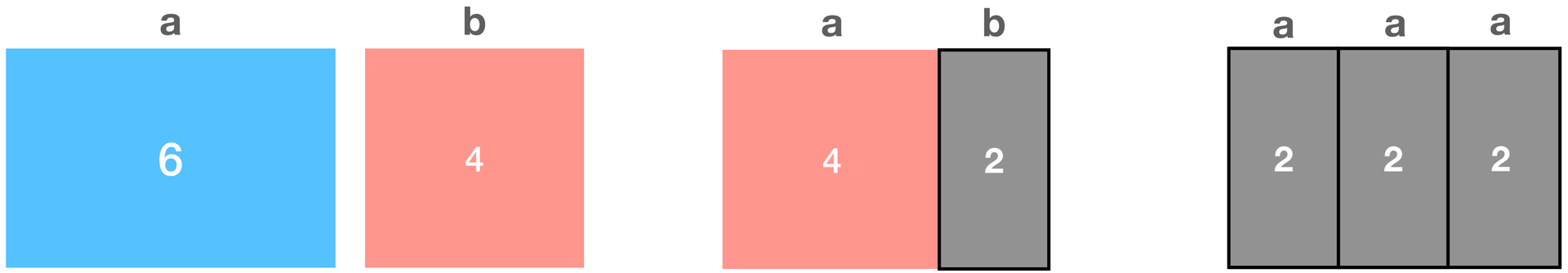
```c
#include <stdio.h>
int main() {
    int i = 1, sum = 0;
    printf("%d", sum);
    while (i <= 1000) {
        sum += i;
        i += 1;
    }
    printf("1到1000的和是：%d\n", sum);
}
```

找最大公因數

- 使用「輾轉相除法」，又稱歐幾里得演算法
- 有兩個整數a和b (以a > b為例)
  - 步驟 1：計算 a 除以 b 的餘數 temp（即 temp = a % b）。
  - 步驟 2：將 a 更新為 b，將 b 更新為 temp。
  - 重複步驟 1 和 2，直到 b 等於 0。
  - 結束：當 b 等於 0 時，a 就是兩數的 GCD。

找最大公因數

找6和4的最大公因數



舊的值	a=6	b=4	a=6	b=4	a=4	b=2
新的值			a=4	b=2	a=2	b=0

使用while迴圈找最大公因數

```c
#include <stdio.h>
int main(){
    int a, b, temp;
    scanf("%d %d", &a, &b);
    while (b != 0) {
        temp = a % b;
        a = b;
        b = temp;
    }
    printf("最大公因數為%d", a);
}
```

for 和 while 的使用 (1)

```c
#include <stdio.h>
int main() {
    int sum = 0;
    for (int i = 1; i <= 1000; i++) {
        sum += i;
    }
    printf("%d", sum);
}
```

```c
#include <stdio.h>
int main() {
    int i = 1, sum = 0;
    printf("%d", sum);
    while (i <= 1000) {
        sum += i;
        i += 1;
    }
    printf("%d", sum);
}
```

設定變數初始值

條件限制

變數值增減

for 和 while 的使用 (2)

- 共同點：
  - 檢查條件是否成立後進入迴圈
- 不同點：
  - while 需要自行維護條件變數 (例如要在迴圈中自行變更i的值才能控制迴圈)
  - 我們不一定知道while會執行幾次

while 搭配 scanf

```c
#include <stdio.h>
int main() {
    int target = 7;  // 正確的數字
    int guess;

    printf("猜一個數字：\n");
    scanf("%d", &guess);

    while (guess != target) {
        if (guess < target) {
            printf("太小了！再試一次：\n");
        } else {
            printf("太大了！再試一次：\n");
        }
        scanf("%d", &guess);
    }
    printf("恭喜！你猜對了！\n");
}
```

C語言主要迴圈結構

- for
- while
- do-while

do while 迴圈介紹

```
do {
    迴圈敘述主體;
} while(進入下一次迴圈的條件);
```

- do-while 的結構和for與while不同，是先執行迴圈敘述主體，再進行條件判斷

do while 範例

```c
#include <stdio.h>
int main() {
    int sum = 0;
    int i = 1;
    do {
        sum += i;  // 每次將 i 加入總和
        i++;
    } while (i <= 1000);  // 當 i 小於等於 1000 時繼續迴圈
    printf("1 到 1000 的總和是: %d", sum);
}
```

while vs. do-while

```c
#include <stdio.h>
int main() {
    int sum = 0;
    int i = 1000;
    do {
        sum += i;
        i++;
    } while (i < 1000);
    printf("%d", sum);
}
```

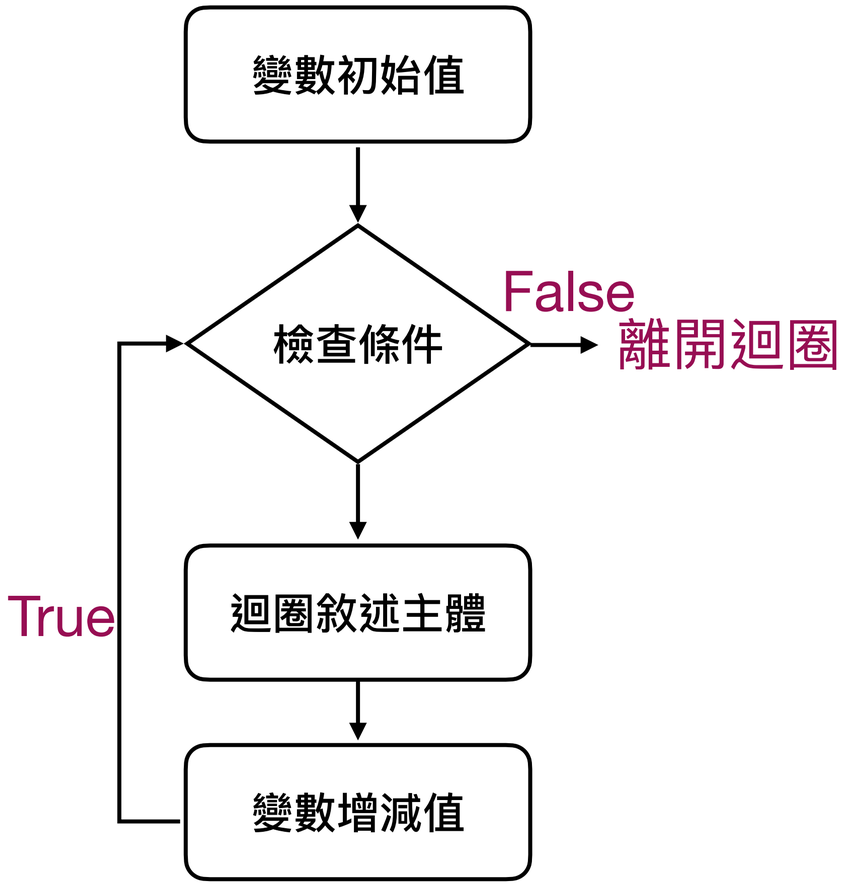
會執行一次迴圈敘述，sum為1000

```c
#include <stdio.h>
int main() {
    int sum = 0;
    int i = 1000;
    while (i < 1000){
        sum += i;
        i++;
    }
    printf("%d", sum);
}
```

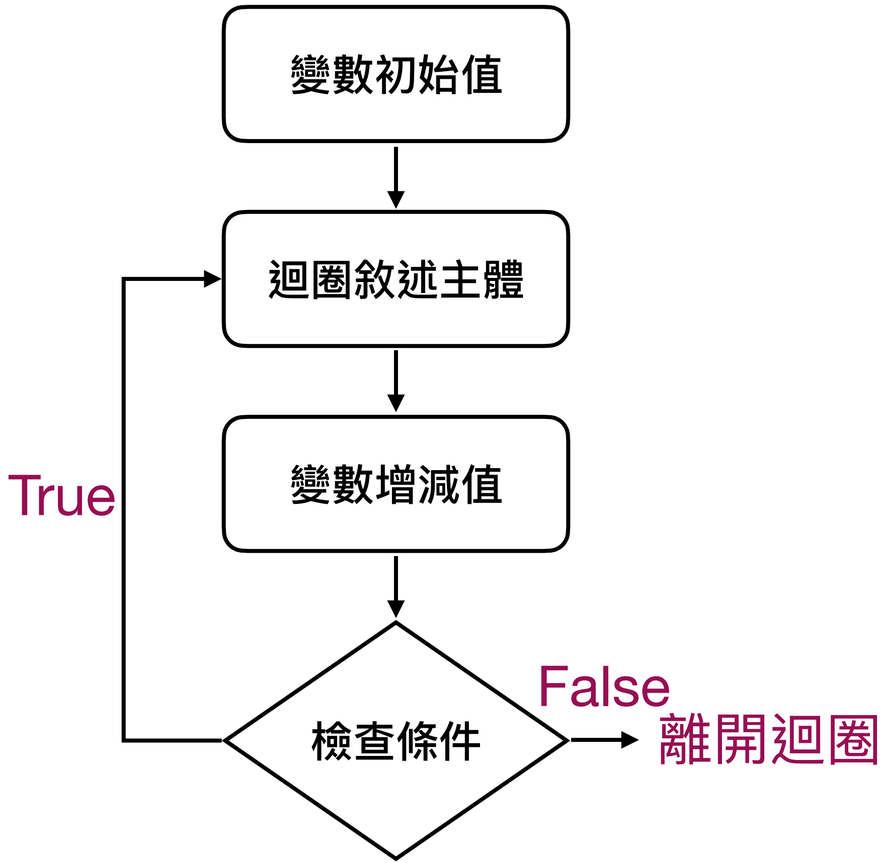
不會執行迴圈敘述

while 與 do-while 流程圖比較

while



do-while



變數範圍 (Scope)

```c
#include <stdio.h>
int main() {
    int sum = 0;
    for (int i = 1; i <= 1000; i++) {
        sum += i;
    }
    printf("%d", i);
}
```

- 此程式將導致編譯失敗，因為 i 只有在 for 迴圈中被定義，故無法被印出
  - 此程式中，i 的變數範圍（或稱「作用域」）只存在於 for 迴圈之內

強迫進入for/while迴圈

```c
int num;
while (1) {  // 這將進入無限迴圈
    printf("現在數字為: %d\n", num);
    num++;
}
```

```c
int num;
for (;;) {       // 這也將進入無限迴圈
    printf("現在數字為: %d\n", num);
    num++;
}
```

強迫進入while迴圈來連續取得scanf

```c
#include <stdio.h>
int main() {
    int score, sum = 0, count = 0;
    float average;
    while (1){
        if (scanf("%d", &score) == EOF)
            break;
        else {
            sum += score;
            count++;
        }
    }
    printf("平均成績: %f", (float)sum / count);
}
```

scanf 說明

- scanf 在讀取不到資料的時候會回傳EOF
- EOF被表示為一個整數形式，值為 -1
  - https://cplusplus.com/reference/cstdio/EOF/