CHAPTER

6

C 語言遞迴

6.1 遞迴

函式間可以相互呼叫，除了呼叫別的函式外，也可呼叫自己本身，這種呼叫自己方式的函式稱為「遞迴」。遞迴是一種應用極廣的程式設計技術，在函式執行的過程不斷的呼叫函式自身，但每一次呼叫，皆會產生不一樣的結果，直到遇到終止再呼叫函式自身的條件或結果時，才會停止遞迴逐次離開函式。如果遞迴的函式內沒有設定終止呼叫的條件，這樣的函式會形成無窮遞迴。

一個問題如果能拆成同形式且較小範圍時，就可以使用遞迴函式來設計。例如要計算 1 + 2 + … + 10 的總和時，可以拆成 1 和 2 + 3 + … + 10，而 2 + 3 + … + 10 又可以拆成 2 和 3 + 4 + … + 10，其餘類推，此時就可以設計成遞迴函式。遞迴函式常使用在具有規則性的程式設計中，其優點是具結構化可以增加程式的可讀性，以及能以簡潔的程式處理反覆的複雜問題。

遞迴在數學或電玩遊戲上常被使用，例如：數列、階乘、費氏數列、輾轉相除法、排列、組合、堆疊、河內塔、八個皇后、老鼠走迷宮…等。有些程式雖然使用 for、while… 等重複結構也能處理，但使用遞迴函式，程式碼會較為簡潔，本章將針對遞迴的基本範例在設計上做詳細的說明。

6.2 數列

本節提供兩個數列函式求總和的範例，說明如何使用遞迴解題的方式來撰寫程式。如下：

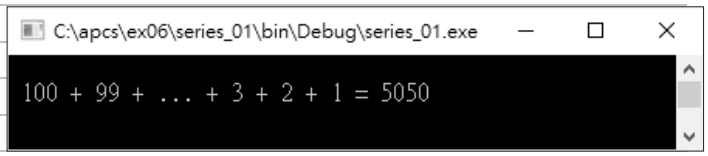
1. sum = n + (n-1) + ⋯ + 3 + 2 + 1
2. sum = 1 – 4 + 7 – 10 – ⋯ + (n-3) – n 或 1 – 4 + 7 – 10 – ⋯ – (n-3) + n

範例 ：series_01.c

使用遞迴函式計算 n + (n-1) + (n-2) + … + 3 + 2 + 1 的結果，其中 n = 100。

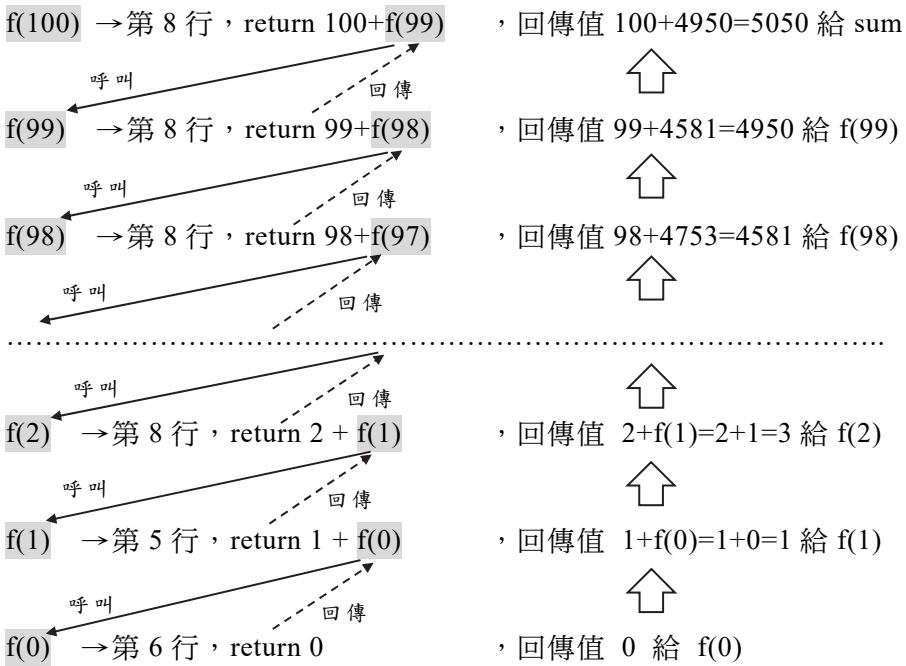
程式碼 FileName : series_01.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int f(int n) {
05     if (n <= 0)
06         return 0;
07     else                    // n > 0
08         return n + f(n-1);
09 }
10
11 int main()
12 {
13     int n = 100, sum;
14     sum = f(n);
15     printf("\n %d + %d + ... + 3 + 2 + 1 = %d", n, n-1, sum);
16     printf("\n") ;
17     return 0;
18 }
```



說明

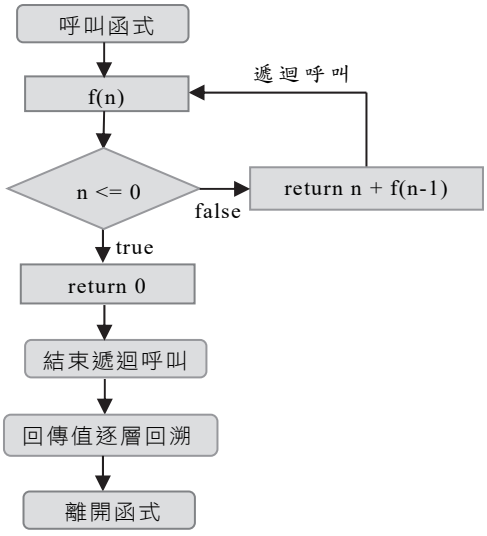
1. 第 4~9 行：建立 f(n) 遞迴函式。該函式被呼叫 f(100) 的流程如下所示：



2. 第 14 行：將 f(100) 的回傳值 5050 指定給 sum 變數。

3. 遞迴函式一定要有結束遞迴的敘述。當第 5 行的條件式 (n<=0) 成立時，執行第 6 行 return 0，就不再遞迴呼叫，而將回傳值逐層回溯給原呼叫敘述。

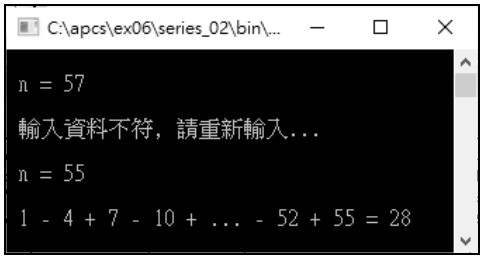
4. 遞迴函式的流程圖如右：

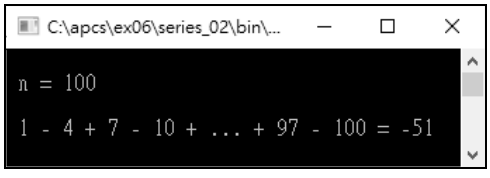


範例 ：series_02.c

使用遞迴函式計算 1 – 4 + 7 – 10 + 13 – … n 的結果，其中 n 由使用者輸入。n 的輸入值必須符合 (n % 3 == 1) 條件，即 n 為 3 的倍數加 1，如 1、4、7、10、14…等。

執行結果





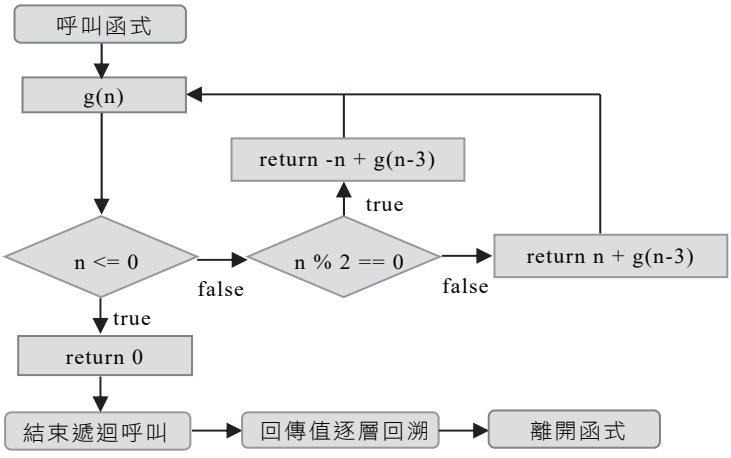
程式碼 FileName : series_02.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int g(int n) {
05     if (n <= 0) {
06         return 0;
07     } else if (n % 2 == 0) {        // n > 0,為偶數
08         return -n + g(n-3);
09     } else {                        // n > 0,為奇數
```

```
10        return n + g(n-3);
11    }
12 }
13
14 int main()
15 {
16    int n, sum;
17    while (1) {
18        printf("\n n = ");
19        scanf("%d", &n);
20        if (n % 3 == 1)
21            break;              //輸入值符合條件跳離迴圈
22        else
23            printf("\n 輸入資料不符, 請重新輸入...\n");
24    }
25
26    sum = g(n);
27    if (n % 2 == 1)     // n 輸入值為奇數
28        printf("\n 1 - 4 + 7 - 10 + ... - %d + %d = %d", n-3, n, sum);
29    else                // n 輸入值為偶數
30        printf("\n 1 - 4 + 7 - 10 + ... + %d - %d = %d", n-3, n, sum);
31
32    printf("\n\n");
33    return 0;
34 }
```

說明

1. 遞迴函式的流程圖如下所示：



2. 第 4~12 行：建立 g(n) 遞迴函式。該函式被呼叫 g(100) 的流程如下：

   g(100)
   → return -100 + g(97)
   → -100 + return 97 + g(94)
   → -100 + 97 + return -94 + g(91)
   ……
   → -100 + 97 – 94 + … -10 + return 7 + g(4)
   → -100 + 97 – 94 + … -10 + 7 + return -4 + g(1)
   → -100 + 97 – 94 + … -10 + 7 - 4 + return 1 + g(0)
   → -100 + 97 – 94 + … -10 + 7 - 4 + 1 + 0
   → -51 (回傳值)

3. 第 20~23 行：篩選使用者的輸入值是否符合 (n % 3 == 1) 的條件。
4. 第 26 行：呼叫 g(n) 的遞迴計算結果回傳指定給 sum 變數。
5. 第 27,28 行：若輸入值為奇數時，印出遞迴函式執行的加減過程，其中最後兩數是先減後加。
6. 第 29,30 行：若輸入值為偶數時，印出遞迴函式執行的加減過程，其中最後兩數是先加後減。

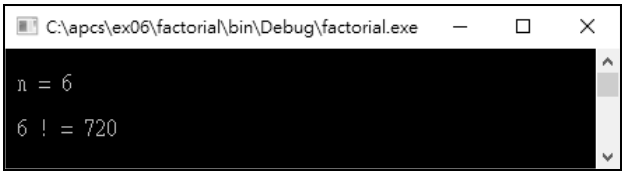
6.3 階乘

在數學中，正整數的「階乘」是所有小於及等於該數 n 的正整數的乘積，以 n! 表示。階乘的公式為 n! = n * (n-1) * (n-2) * … * 3 * 2 * 1，例如：
5! = 5 * 4 * 3 *2 * 1 = 120。

範例 ：factorial.c

使用階乘函式計算 n! = 1 * 2 * 3 * (n-1) * n 的結果，其中 n 由使用者輸入。
n 的輸入值必須大於等於 1。

執行結果



程式碼 FileName : factorial.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int d(int n){
05     if (n <= 1) {
06         return 1;
07     } else {                    // n > 1
08         return n * d(n-1);
09     }
10 }
11
12 int main()
13 {
14     int n, fac;
15     while (1) {
16         printf("\n n = ");
17         scanf("%d",&n);
18         if (n >= 1)
19             break;
20         else
21             printf("\n 輸入資料不符, 請重新輸入... \n");
22     }
23     fac = d(n);
24     printf("\n %d ! = %d", n, fac);
25
26     printf("\n\n");
27     return 0;
28 }
```

說明

1. 第 4~10 行：建立 d(n) 遞迴函式。該函式被呼叫 d(6) 的流程如下所示：

   d(6)

   → return 6 * d(5)

   → return 6 * 5 * d(4)

   → return 6 * 5 * 4 * d(3)

   → return 6 * 5 * 4 * 3 * d(2)

   → return 6 * 5 * 4 * 3 * 2 * d(1)

   → return 6 * 5 * 4 * 3 * 2 * 1 → return 720 (回傳值)

2. 第 18~21 行：篩選使用者的輸入值是否符合 (n >= 1) 的條件。

3. 第 23 行：呼叫 d(6) 的遞迴計算結果，回傳指定給 fac 變數。

題目 (5)

給定右側程式，當程式執行完後，

輸出結果為何？

(A) 1 2 3 4 5 6 7 8

(B) 7 5 3 1 2 4 6 8

(C) 7 5 3 2 1 4 8 6

(D) 8 7 6 5 4 3 2 1

```
int A[8] = {8,7,6,5,4,3,2,1};
int main() {
   int i, j;
   for (i=0; i<8; i=i+1) {
      for(j=i; j<7; j=j+1) {
         if(A[j] > A[j+1]) {
            A[j] = A[j] + A[j+1];
            A[j+1] = A[j] - A[j+1];
            A[j] = A[j] - A[j+1];
         }
      }
   }
   for (i=0; i<8; i=i+1) {
      printf("%d ", A[i]);
   }
}
```

說明

1. A[8] = {8,7,6,5,4,3,2,1}
   當 i=0、j=0 時，因 (A[0]=8) > (A[1]=7) 成立，所以
   A[0] = A[0] + A[1] = 8+7 = 15
   A[1] = A[0] – A[1] = 15-7 = 8
   A[0] = A[0] – A[1] = 15-8 = 7
   結果 A[8] = {7,8,6,5,4,3,2,1}。
   當 i=0、j=1 時，因 (A[1]=8) > (A[2]=6) 成立，結果 A[8] = {7,6,8,5,4,3,2,1}。
   當 i=0、j=2 時，結果 A[8] = {7,6,5,8,4,3,2,1}。
   當 i=0、j=3 時，結果 A[8] = {7,6,5,4,8,3,2,1}。
   …….
   當 i=0、j=6 時，結果 A[8] = {7,6,5,4,3,2,1,8}。
   當 i=1、j=1 時，結果 A[8] = {7,5,6,4,3,2,1,8}。
   當 i=1、j=2 時，結果 A[8] = {7,5,4,6,3,2,1,8}。
   ……..
   當 i=1、j=5 時，結果 A[8] = {7,5,4,3,2,1,6,8}。
   當 i=2、j=2 時，結果 A[8] = {7,5,3,4,2,1,6,8}。
   當 i=2、j=3 時，結果 A[8] = {7,5,3,2,4,1,6,8}。
   當 i=2、j=4 時，結果 A[8] = {7,5,3,2,1,4,6,8}。
   當 i=3、j=3 時，結果 A[8] = {7,5,3,1,2,4,6,8}。
2. 之後因元素值都是遞增排列 A[j] > A[j+1] 條件不會成立，所以 A 陣列元素值不變，因此答案是(B)，程式檔請參考 test_05.c。

題目 (6)

給定右側函式 f( )，已知 f(14)、f(10)、f(6) 分別回傳 25、18、10，函式中的 ? 應為下列何者？

(A) (n+1) / 2　　(B) n / 2

(C) (n-1) / 2　　(D) (n/2) + 1

```
int f(int n) {
   if(n < 2) {
      return n;
   }
   else {
      return (n + f( ? ));
   }
}
```

說明

1. 答案是(B)，程式檔請參考 test_06.c。
2. 分別用各選項代入，選項(B) n / 2 符合 f(14)、f(10)、f(6) 分別回傳 25、18、10：
   f(14) = 14 + f(7) = 14 + 7 + f(3) = 14 + 7 + 3 + f(1) = 14+7+3+1 = 25
   f(10) = 10 + f(5) = 10 + 5 + f(2) = 10 + 5 + 2 + f(1) = 10+5+2+1 = 18
   f(6) = 6 + f(3) = 6 + 3 + f(1) = 6 + 3 + 1 = 10

題 目 (7)

給定右側程式，當程式執行完後，輸出結果為何？

(A) 1　　(B) 2

(C) 3　　(D) 4

```
int main()
{
    int a[5] = {9, 4, 3, 5, 3};
    int b[10] = {0,1,0,1,0,1,0,1,0,1};
    int c = 0;
    for (int i=0; i<5; i=i+1)
        c = c + b[a[i]];
    printf("%d,", c);
    return 0;
}
```

說明

1. 答案是(D)，程式檔請參考 test_07.c。
2. 當 i=0 時，因 a[0] = 9，所以 c = c + b[a[0]] = 0 + b[9] = 0 + 1 = 1。
   當 i=1 時，因 a[1] = 4，所以 c = c + b[a[1]] = 1 + b[4] = 1 + 0 = 1。
   當 i=2 時，因 a[2] = 3，所以 c = c + b[a[2]] = 1 + b[3] = 1 + 1 = 2。
   當 i=3 時，因 a[3] = 5，所以 c = c + b[a[3]] = 2 + b[5] = 2 + 1 = 3。
   當 i=4 時，因 a[4] = 3，所以 c = c + b[a[4]] = 3 + b[3] = 3 + 1 = 4。

題 目 (8)

給定右側程式，當程式執行完後，輸出結果為何？

(A) 9

(B) 18

(C) 27

(D) 30

```
01 int Q[200];
02 int i, val=0;
03 int count=0;
04 int head=0, tail=0;
05 for(i=1; i<=30; i=i+1) {
06     Q[tail] = i;
07     tail = tail+1;
08 }
09 while (tail > head+1){
10     val = Q[head];
11     head = head+1;
12     count = count+1;
13     if(count == 3) {
14         count=0;
15         Q[tail] = val;
16         tail = tail+1;
17     }
18 }
19 printf("%d", Q[head]);
```

說明

1. 答案是(B)，程式檔請參考 test_08.c。
2. 執行第 02~08 行後，head=0, Q[0]=1, Q[1]=2, Q[2]=3, … , Q[29]=30, tail=30。

CHAPTER

11 Python 串列

11.1 何謂串列

串列是儲存資料的容器，它是一組經過編排號碼順序的變數，並在記憶體中占用連續的位址空間。如抽屜收納櫃，每個抽屜都依順序標記連續號碼，若要從某個抽屜取得或放置物品，只要知道抽屜是在第幾層，就能很快地找到該抽屜。

若將收納櫃比喻為一個「串列」(list)，則收納櫃的一個個抽屜，在串列上我們稱之為「元素」(element) 。同樣地欲存取串列中某個元素資料的內容，只要告知該元素在串列中的「索引」(index)(也就是編號)，即可存取該元素內容。

11.2 一維串列

若串列中只有一組索引(編號)，稱為一維串列；有兩組索引，稱為二維串列；有三組索引，稱為三維串列；以此類推…。

一. 一維串列的建立

串列建立時，要指定串列的名稱、標示擁有多少個元素。其建立方式是以 [ ] 運算子來存放元素資料，元素間用逗號分隔。其建立的方式如下：

語法

```
串列名稱 = [元素 1, 元素 2, 元素 3, …]
```

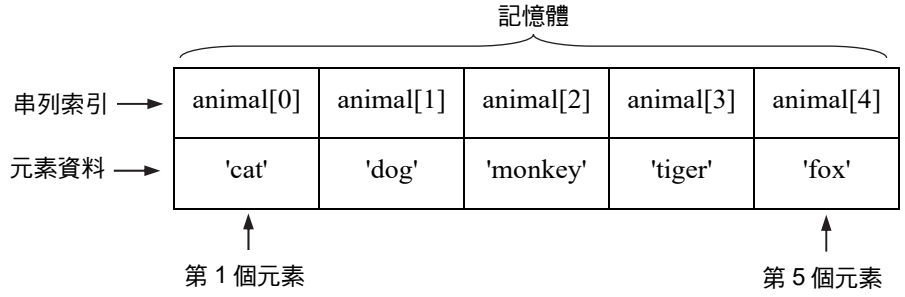
簡例 建立動物 animal 的串列名稱，其元素內容分別為動物名稱的英語單字字串。

```
animal = ['cat', 'dog', 'monkey', 'tiger', 'fox']      # 宣告建立 animal 串列
```

串列名稱　　元素內容

說明

1. animal 是一個串列，擁有 5 個字串元素，依序為：animal[0]、animal[1]、animal[2]、animal[3]、animal[4]。
2. 串列大小為串列長度，由「索引」來編排元素順序，索引由 0 開始。若串列大小為 5，則索引範圍會是 0~4。索引必須是整數字面值、整數變數或整數運算式。
3. animal 串列的每一個元素皆可視為一個變數，這些變數在主記憶體中占用連續位址空間，裡面存放著字串型別的資料，如下圖所示：



串列中各個元素資料的型別可以相同，也可以不相同。

簡例

```
01 score = [62, 88, 89, 73, 100]                  # 元素皆為整數資料型別
02 fruit = ['apple', 'banana', 'lemon', 'pear']    # 元素皆為字串資料型別
03 data = ['John', 36, True]          # 元素存放的資料型別不相同
04 lst = []                           # 空串列,不含元素資料
```

二. 串列的讀取

利用 [ ] 運算子填入串列索引，便可讀取串列對應的元素內容。語法如下：

語法

```
串列名稱 = [索引]
```

說明

1. 串列中第一個元素的索引是 0，第二個元素的索引是 1，以此類推…。
2. 若索引為負數，則從串列尾端倒數來讀取串列元素，串列倒數第一個元素的索引是 -1，倒數第二個元素的索引是 -2。
3. 索引不能超過串列長度範圍，否則程式執行時會產生錯誤。

簡例 建立串列 data，並進行單一元素的讀取操作。

```
01 data = [11, 22, 33, 44, 55, 66, 77, 88, 99]
```

```
02 print(data[0])      # 輸出 11
03 print(data[3])      # 輸出 44
04 print(data[-1])     # 輸出 99
05 print(data[-9])     # 輸出 11
06 print(data[10])     # 錯誤，索引超過範圍
```

若串列是使用 start:end:step 填入[ ] 運算子，便是讀取索引從 strat 起至 end 前一位，間隔 step 之間的串列元素。語法如下：

語法

```
串列名稱 = [start:end:step]
```

說明

1. 讀取索引 strat 起到 end 前一位之間的串列元素，step 為間隔值。
2. 若 strat 省略，預設值為 0；若 end 省略，則預設值為串列長度；若 step 省略，則預設值為 1。

簡例 建立串列 lst，並進行各種不同元素的讀取操作。

```
01 lst = [11, 22, 33, 44, 55, 66, 77, 88, 99]
02 print(lst[3:7])          # 輸出 [44, 55, 66, 77]
03 print(lst[:5])           # 輸出 [11, 22, 33, 44, 55]
04 print(lst[0:9])          # 輸出 [11, 22, 33, 44, 55, 66, 77, 88, 99]
05 print(lst[1:9:2])        # 輸出 [22, 44, 66, 88]
06 print(lst[1::2])         # 輸出 [22, 44, 66, 88]
07 print(lst[-5:-1])        # 輸出 [55, 66, 77, 88]
08 print(lst[:-2])          # 輸出 [11, 22, 33, 44, 55, 66, 77]
09 print(lst)               # 輸出 [11, 22, 33, 44, 55, 66, 77, 88, 99]
```

三. 串列的存放

除空串列外，串列在建立時已存放有元素的初值，但如同變數一樣，所存放的元素值可以再改變內容，其改變元素內容的語法為：

語法

```
串列名稱[索引] = 資料
```

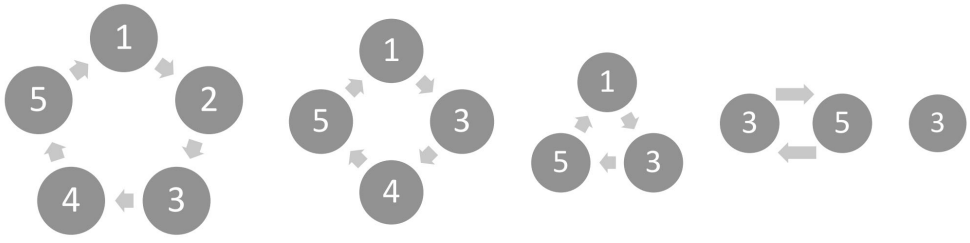
簡例 建立串列 arr，並進行相關的串列元素存放操作。

```
01 arr = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']
02 arr[2] = 'Tuesday'       # arr[2]的內容由'Tue'更改為'Tuesday'
03 print(arr[2])            # 輸出 Tuesday
04 print(arr)               # 輸出 ['Sun', 'Mon', 'Tuesday', 'Wed', 'Thu', 'Fri', 'Sat']
```

14.3 定時 K 彈

問題描述

「定時 K 彈」是一個團康遊戲，N 個人圍成一圈，由 1 號依序到 N 號，從 1 號開始依序傳遞一枚玩具炸彈，每次到第 M 個人就會爆炸，此人即淘汰，被淘汰的人要離開圓圈，然後炸彈再從該淘汰者的下一個開始傳遞。遊戲之所以稱 K 彈是因為這枚炸彈只會爆 K 次，在第 K 次爆炸後，遊戲即停止，而此時在第 K 個淘汰者的下一位遊戲者被稱為幸運者，通常就會要求表演節目。例如 N=5，M=2，如果 K=2，炸彈會爆兩次，被爆炸淘汰的順序依是 2 與 4 (參見下圖)，這時 5 號就是幸運者。如果 K=3，剛才的遊戲會繼續，第三個淘汰是 1 號，所以幸運者是 3 號。如果 K=4，下一輪淘汰 5 號，所以 3 號是幸運者。



輸入格式

輸入只有一行包含三個正整數，依序為 N、M 與 K，兩數中間有一個空格分開。其中 $1 \le K<N$。

輸出格式

請輸出幸運者的號碼，結尾有換行符號。

範例一：輸入

```
5 2 4
```

範例一：正確輸出

```
3
```

【說明】
被淘汰的順序是 2、4、1、5，此時5的下一位是3，也是最後剩下的，所以幸運者是3。

範例二：輸入

```
8 3 6
```

範例二：正確輸出

```
4
```

【說明】
被淘汰的順序是 3、6、1、5、2、8，此時8的下一位是4，所以幸運者是4。

評分說明

輸入包含若干筆測試資料，每一筆測試資料的執行時間限制 (time limit) 均為 1 秒，依正確通過測資筆數給分。其中：

第 1 子題組 20 分，1 ≤ N ≤ 100，且 1 ≤ M ≤10，K = N - 1。

第 2 子題組 30 分，1 ≤ N ≤ 10,000，且 1 ≤ M ≤1,000,000，K = N - 1。

第 3 子題組 20 分，1 ≤ N ≤ 200,000，且 1 ≤ M ≤ 1,000,000，K = N - 1。

第 4 子題組 30 分，1 ≤ N ≤ 200,000，且 1 ≤ M ≤ 1,000,000，1 ≤ K < N。

解題分析

1. 本題以串列來模擬此遊戲的過程，每一個串列元素代表一個玩家，玩家編號由 1 開始，串列索引值是由 0 開始，所以若有 N 個玩家，串列索引值會是從 0 到 N-1 排列，每一個串列內儲存的是下一個串列的索引值。例如：串列索引值 0 的串列內容是 1，代表第 1 個玩家的下一位玩家是 2 號。由於參與遊戲的人員是圍成一個圓圈，所以 N-1 的串列內容是 0，如此串列便頭尾相接，形成一個環狀。

```
lstMan = list(range(1, N + 1))
pre = N - 1
lstMan[pre] = 0
point = 0
```

2. 遊戲從玩家 1 開始，其串列索引值是 0，其上家是玩家 5，索引值是 4，所以 point=0 (point 變數紀錄目前玩家的索引值)，pre=4 (pre 變數紀錄上家的索引值)。

3. 下表呈現的是範例一的遊戲過程：

玩家編號	1	2	3	4	5	point	pre
串列索引值	0	1	2	3	4		
初始值	1	2	3	4	0	0	4
1-1	1	2	3	4	0	1	0
1-2	1→2	2	3	4	0	2	0
2-1	2	2	3	4	0	3	2
2-2	2	2	3→4	4	0	4	2
3-1	2	2	4	4	0	0	4
3-2	2	2	4	4	2	2	4
4-1	2	2	4	4	2	4	2
4-2	2	2	4→2	4	2	2	2

① 遊戲過程中 m 值如果大於零 (第 03 行)，則炸彈不會引爆，遊戲流程繼續進行，目前玩家成為上家 (第 04 行)，下一個玩家成為新玩家 (第 05 行)，m 值遞減 1 (第 06 行)。

② 遊戲過程中，如果炸彈引爆 (m 值等於零)，則該玩家將被淘汰，所以該玩家要告知其上家，自己的下一位玩家編號，然後退出遊戲。實作上是以串列 [point] 的內容值複製到串列 [pre] (第 09 行)，同時串列 [point] 的內容值為下一回合的起始 (第 10 行)，此時被淘汰的玩家的索引值不再出現在遊戲串列中。恢復 m 值 (第 07 行)，引爆次數 K 值遞減 1 (第 08 行)。

```
01 m = M
02 while K > 0:
03     while m > 1:
04         pre = point
05         point = lstMan[point]
06         m -= 1
07     m = M
08     K -= 1
09     lstMan[pre] = lstMan[point]
10     point = lstMan[point]
```

4. 以此類推，直到炸彈引爆 K 次為止。此時的 point 加 1 就是幸運者的玩家索引值。

程式碼 FileName : apcs_10510_03.py

```
01 sT = input()
02 lstT = sT.split(' ')
03 N = int(lstT[0])
04 M = int(lstT[1])
05 K = int(lstT[2])
06 lstMan = list(range(1, N + 1))
07 pre = N - 1
08 lstMan[pre] = 0
09 point = 0
10 m = M
11 while K > 0:
12     while m > 1:
13         pre = point
14         point = lstMan[point]
15         m -= 1
16     m = M
17     K -= 1
18     lstMan[pre] = lstMan[point]
19     point = lstMan[point]
```