CHAPTER

2

陣列

2.1 陣列表示法

在未談及陣列(array)之前，讓我們先來看看線性串列(linear list)。線性串列又稱循序串列(sequential list)或有序串列(ordered list)。其特性是每一項資料是依據它在串列的位置，所形成的一個線性排列，所以 x[i]會出現在 x[i+1]之前。

線性串列經常的操作如下：

1. 取出串列中的第 i 項的資料；$0 \le i \le n-1$。
2. 計算串列的長度。
3. 由左至右或由右至左取出此串列的資料。
4. 在第 i 項加入一個新值，使其原來的第 i，i+1，…，n 項變為第 i+1，i+2，…， n+1 項。亦即在 i 之後的資料都要退後一個位址。
5. 刪除第 i 項，使其原來的第 i+1，i+2，……，n 項變為第 i，i+1，……，n–1 項。亦即在 i 之後的資料都會往前一個位址。

在 Java 程式語言中，我們利用陣列來表示線性串列，以線性的對應方式將元素 a_i 置於陣列的第 i 個位置上，若要讀取 a_i 時，可利用 $a_i = a_0 + i*d$ 求得。其中 a_0 為陣列的起始位址，d 為每一個元素所佔的空間大小。要注意的是，Java 陣列的註標是從 0 開始。

以下將介紹陣列的表示法。

2.1.1 一維陣列

若一維陣列(one dimension array)是 A(0：u－1)，而且每一個元素佔 d 個空間，則 **$A(i) = a_0 + i*d$**，其中 a_0 是陣列的起始位置。若每一元素，所佔的空間為 d，且起始的元素為 0，則陣列 A 的每一元素所對應的位址表示如下：(假設 d=1)

陣列元素：	A(0)	A(1)	A(2)	…	A(i)	…	A(u–1)
位　　址：	a_0	a_0+1	a_0+2	…	a_0+(i)	…	a_0+(u–1)

若陣列是 A(t：u)，則 **$A(i) = a_0 + (i–t)*d$**。

如陣列 A(2：12)，則 A(6)與 A(2)起始點相差 4 個單位(6–2)，相當於上述(i–t)。

若陣列為 A(1：u)，表示陣列的起始元素位置從 1 開始，則 $A(i) = a_0 + (i–1)*d$，其中 d 表示每一元素所佔的空間大小。

範例》

1. 有一陣列 A(0：100)，而且 A(0) 的位址是 100，d 為 2，試問 A(16)是多少？

   解 由於 $A(i) = 100 + (i)*d$

   $A(16) = 100 + 16*2 = 132$

2. 有一陣列 A(–3：10)，而且 A(–3)=100，d=1，求 A(5)=？

   解 由於陣列的形式是(t：u)，所以

   $A(i) = a_0 + (i–t)*d$

   $A(5) = 100 + (5–(–3))*1$

   $= 108$

2.1.2 二維陣列

假若有一個二維陣列(two dimension array)是 $A[0：u_1–1, 0：u_2–1]$，表示此陣列有 u_1 列及 u_2 行；也就是每一列是由 u_2 個元素所組成。二維陣列化成一維陣列時，對應的方式有二種：(1)以列為主 (row major)，(2)以行為主 (column major)。

1. **以列為主**：視此陣列有 u_1 個元素 0，1，2，…，$u_1–1$ ，每一元素有 u_2 個單位，每個單位佔 d 個空間。其情形如圖 2-1 所示：

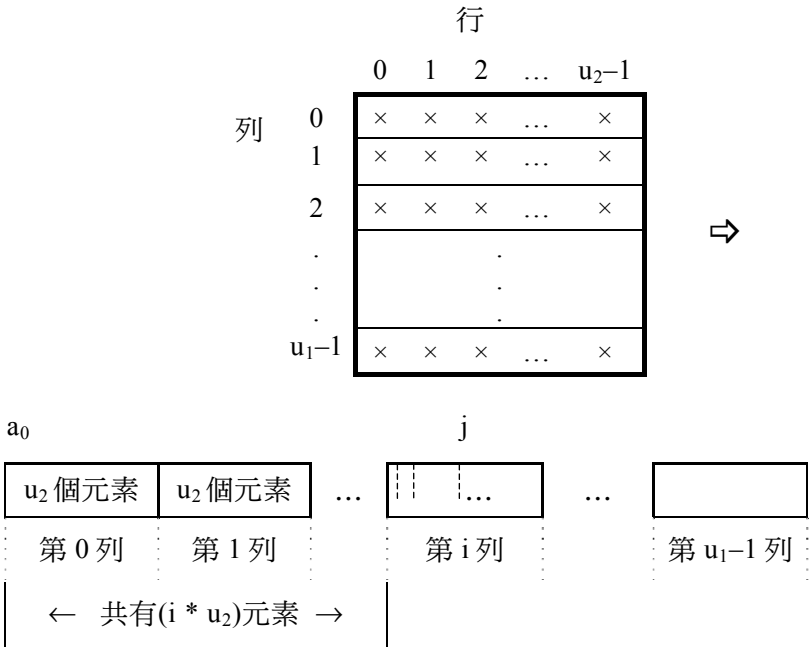


圖 2-1 以列爲主的二維陣列循序表示

由上圖可知 **A(i, j) = a_0 + i*u_2*d + j*d**

2. **以行為主**：視此陣列有 u_2 個元素 1，2，…，u_2，每一元素有 u_1 個單位，每個單位佔 d 個空間。其情形如圖 2-2 所示：

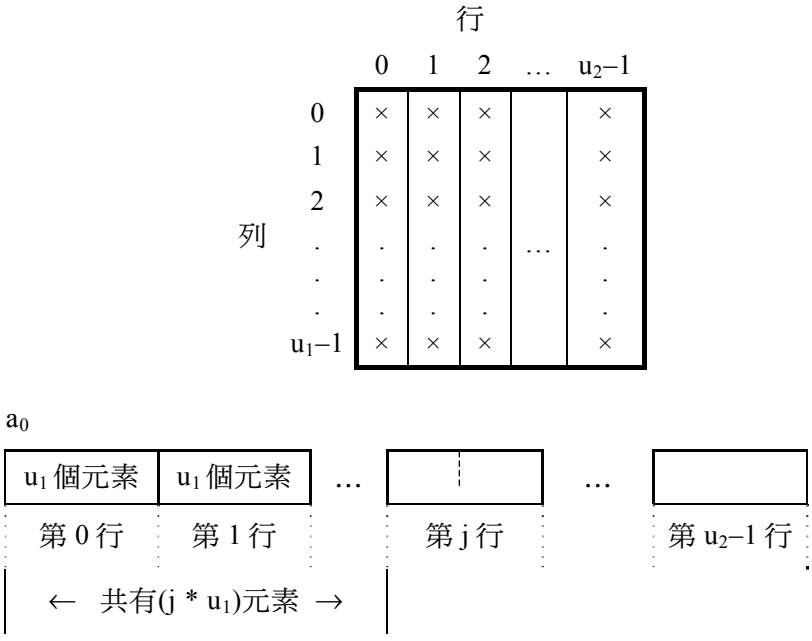


圖 2-2 以行爲主的二維陣列循序表示

由上圖可知 **$A(i, j) = a_0 + j*u_1*d + i*d$**

假若陣列是 $A(s_1 : u_1, s_2 : u_2)$，則此陣列共有 $m = u_1 - s_1 + 1$ 列，$n = u_2 - s_2 + 1$ 行。計算 A(i, j)的位址如下：

1. 以列為主

   $A(i, j) = a_0 + (i-s_1)*n*d + (j-s_2)d$

2. 以行為主

   $A(i, j) = a_0 + (j-s_2)*m*d + (i-s_1)d$

範例》

假設二維陣列為 $A(-3 : 5, -4 : 2)$，起始位址是 $A(-3, -4) = 100$，而且是以列為主排列，請問 A(1, 1)所在的位址？(d=1)

解 $m = 5 - (-3) + 1 = 9$, $n = 2 - (-4) + 1 = 7$, $s_1=-3$, $s_2=-4$, $i=1$, $j=1$

$A(i, j) = a_0 + (i-s_1)*n*d + (j-s_2)d$

$A(1, 1) = 100 + (1-(-3))*7*1 + (1-(-4))1$

$= 100 + 4*7 + 5$

$= 133$

另一解法是將其化為標準式

$A(-3 : 5, -4 : 2) \rightarrow A(0 : 8, 0 : 6)$，得知 $u_1=9$，$u_2=7$

$A(-3, -4) \rightarrow A(0, 0)$，得知 $A(1, 1) \rightarrow A(4, 5)$

$\therefore A(4, 5) = 100 + 4 \times 7 + 5 = 100 + 28 + 5 = 133$

2.1.3 三維陣列

假若有一個三維陣列(three dimension array)是 $A(0 : u_1-1, 0 : u_2-1, 0 : u_3-1)$，如圖 2-3 所示：

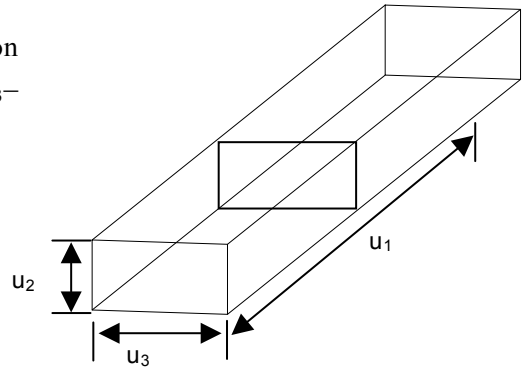
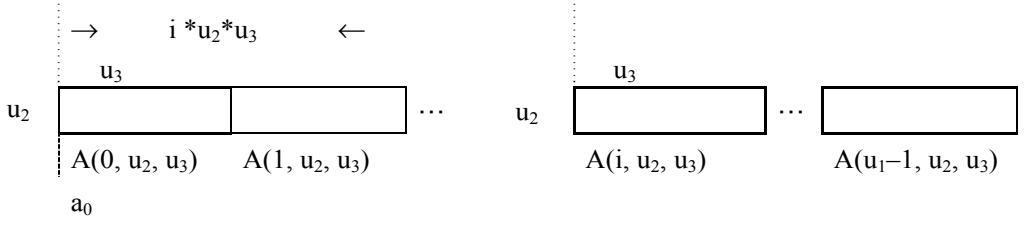


圖 2-3 三維陣列以 u_1 個二維陣列來表示

一般三維陣列皆先轉為二維陣列後，再對應到一維陣列，對應方式也有二種：(1)以列為主，(2)以行為主。

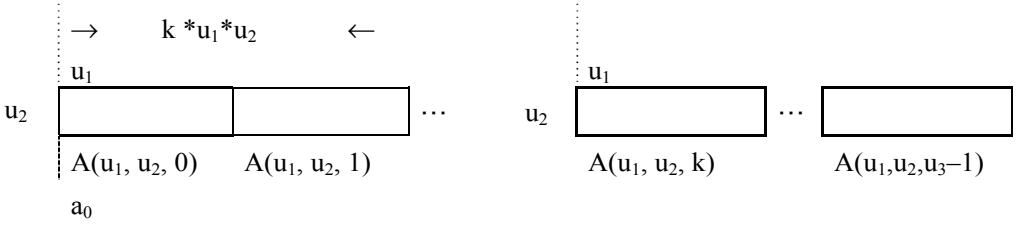
1. **以列為主**

視此陣列有 u_1 個 u_2*u_3 的二維陣列，每一個二維陣列有 u_2 個元素，每個 u_2 皆有 u_3d 個空間。



$A(i , j , k) = a_0 + i*u_2*u_3*d + j*u_3*d + k*d$

2. **以行為主**



$A(i , j , k) = a_0 + k*u_1*u_2*d + j*u_1*d + i*d$

假設陣列為 $A(s_1：u_1, s_2：u_2, s_3：u_3)$，則 $p = u_1 – s_1+1$，$q = u_2 – s_2+1$，$r = u_3 – s_3+1$。

以列為主的公式為：

$A(i , j , k) = a_0 + (i–s_1)*q*r*d + (j–s_2)*r*d + (k–s_3) *d$

以行為主的公式為：

$A(i , j , k) = a_0 + (k–s_3)*p*q*d + (j–s_2)*p*d + (i–s_1) *d$

範例》

假設有一三維陣列 A(–3：5, –4：2, 1：5)，其起始位址為 A(–3, –4, 1) = 100，且是以列為主的排列，試求 A(1, 1, 3)所在的位址？(d=1)

解 $p = 5– (–3) + 1 = 9$, $q = 2–(–4) + 1 = 7$, $r = 5 – 1 + 1 = 5$,

$s_1=–3$, $s_2=–4$, $s_3=1$; $i=1$, $j=1$, $k=3$

$A(1, 1, 3) = 100 + (1–(–3)) * 7*5*1 + (1–(–4)) *5*1 + (3–1)*1 = 267$

另一種計算方式是為將式子化為標準式來看

A(–3：5, –4：2, 1：5)→A(0：8, 0：6, 0：4)　　①

從上式得知 $u_1=9$，$u_2=7$，$u_3=5$

A(1, 1, 3)→A(4, 5, 2)　　②

此乃第①式化為標準式，將(-3, -4, 1)分別加 3，加 4 及減 1 的原故，所以也要對(1, 1, 3) 加 3，加 4 及減 1 的動作，使其成為第②式。同時 A(–3, –4, 1)→A(0, 0, 0)=100 (已知)

$$\therefore A(4,5,2) = 100 + 4*7*5 + 5*5 + 2$$
$$= 100 + 140 + 25 + 2$$
$$= 267$$

此答案與上一種解法所求出的答案是相同。

2.1.4　n 維陣列

假若有一 n 維陣列(n dimension array)為 A(0：u_1–1, 0：u_2–1, 0：u_3–1, … , 0：u_n–1)，表示 A 陣列為 n 維陣列，同樣 n 維陣列亦有二種表示方式：(1)以列為主，(2)以行為主。

1. **以列為主**：若 A 陣列以列為主，表示 A 陣列有 u_1 個 n–1 維陣列，u_2 個 n–2 維陣列，u_3 個 n–3 維陣列，…及 u_n 個一維陣列。假設起始位址為 a_0，則

A(0, 0, 0, … , 0)之位址為	a_0
A(i_1, 0, 0, … , 0)之位址為	$a_0 + i_1 * u_2\ u_3 \cdots u_n$
A(i_1, i_2, 0, … , 0) 之位址為	$a_0 + i_1 * u_2\ u_3 \cdots u_n$
	$+ i_2 * u_3\ u_4 \cdots u_n$
…	
A(i_1, i_2, i_3, … , i_n)之位址為	$a_0 + i_1 * u_2\ u_3 \cdots u_n$
	$+ i_2 * u_3\ u_4 \cdots u_n$
	$+ i_3 * u_4\ u_5 \cdots u_n$
…	
	$+ i_{n-1} * u_n$
	$+ i_n$

上述可歸納為：

$\mathbf{A(i_1, i_2, i_3, \cdots, i_n) = a_0 + \sum_{m=1}^{n} i_m * a_m}$ ，其中

$$\begin{cases} a_m = \prod_{p=m+1}^{n} u_p, 1 \le m < n \\ a_n = 1 \end{cases}$$

(此處的 π 是連乘的意思)

2. **以行為主**：若 A 陣列以行為主，表示 A 陣列有 u_n 個 n–1 維陣列，u_{n-1} 個 n–2 維陣列，…，u_j 個 j–1 維陣列及 u_2 個一維陣列。假設起始位址亦是 a_0，則

$A(0, 0, 0, \cdots, 0)$ 之位址為 a_0

$A(0, 0, 0, \cdots, i_n)$ 之位址為 $a_0 + i_n * u_1 u_2 \cdots u_{n-1}$

$A(0, 0, 0, \cdots, i_{n-1}, i_n)$ 之位址為 $a_0 + i_n * u_1 u_2 \cdots u_{n-1}$
$+ i_{n-1} * u_1 u_2 \cdots u_{n-2}$
…

$A(i_1, i_2, i_3, \cdots, i_n)$ 之位址為 $a_0 + i_n * u_1 u_2 \cdots u_{n-1}$
$+ i_{n-1} * u_1 u_2 \cdots u_{n-2}$
$+ i_{n-2} * u_1 u_2 \cdots u_{n-3}$
…
$+ i_2 * u_1$
$+ i_1$

上述可歸納為：

$A(i_1, i_2, i_3, \cdots, i_n) = a_0 + \sum_{m=1}^{n} i_m * a_m$ ，其中

$$\begin{cases} a_m = \prod_{p=1}^{m-1} u_p, 2 \le m < n \\ a_1 = 1 \end{cases}$$

練習題

1. 有一個二維陣列如下：$A(1：u_1, 1：u_2)$，若分別寫出(a)以列為主(b)以行為主的 A(i, j)=？
2. 假設 $A(-3：5, -4：2)$且其起始位址 $A(-3, -4) = 100$，以行為主排列，請問 A(1, 1)所在的位址？(d=1)

3. 若有一陣列為 A(1：u_1, 1：u_2, 1：u_3)，試問 A(i, j, k)分別以列為主，和以行為主所在的位址各為何？

4. 假設有一個三維陣列 A(–3：5, –4：2, 1：5)，其起始位址為 A(–3, –4, 1)=100，而且是以行為主排列，則 A(2, 1, 2)所在的位址為何？(d=1)

5. 假設有一個 n 維陣列 A(1：u_1, 1：u_2, 1：u_3…, 1：u_n)，試分別寫出以列為主和以行為主的 A(i_1, i_2, i_3, …, i_n) 之位址為何？

2.2 Java 語言的陣列表示法

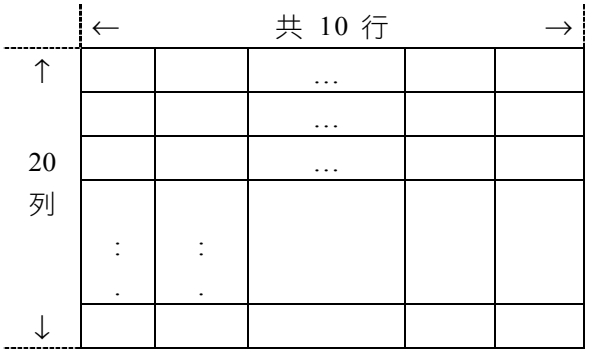
Java 語言的一維陣列表示如下：

```
int A [] = new int[20];  // 或 int[] A = new int[20]
```

表示 A 陣列有 20 個整數元素，從 A[0]到 A[19]，此處陣列的元素乃以大分號表示之。注意！Java 語言的陣列註標起始值為 0，而二維陣列表示方法為

```
int A [][] = new int[20][10];  // 或 int[][] A = new int[20][10]
```

表示 A 陣列有 20 列、10 行，如下圖所示：

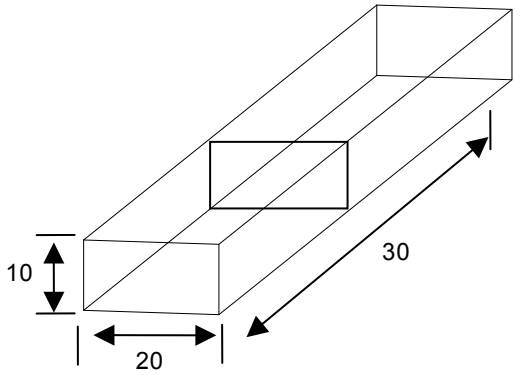


從 A[0][0]，A[0][1]，…，A[19][9]等 200 個元素。

以此類推，三維陣列就是由三個中括號表示之，如

```
int A [][][] = new int[30][20][10];
```

其所表示的圖形就像三度空間，如下圖所示：



此陣列共有 6000 個元素。

練習題

1. 試問下列片段程式的輸出結果為何？

(a)

```
int i[] = new int[10];
int k, total = 0;
for(k=0; k<10; k++)
   i[k] = k+1;
for(k=0; k<10; k++)
   total += i[k];
System.out.printf("%d\n", total);
```

(b)

```
int arr2[][] = new int[5][5];
int i, j, total = 0;
for (i=0; i<5; i++) {
   for (j=0; j<5; j++) {
       arr2[i][j] = i + j;
       System.out.printf("%3d", arr2[i][j]);
   }
   System.out.printf("\n");
}
for (i=0; i<5; i++)
   for (j=0; j<5; j++)
       total += arr2[i][j];
System.out.printf("toal = %d\n", total);
```

2. 將第 1 章動動腦時間的第 2 題，實際以 Java 程式來執行，並檢查你做的答案是否與它相符合。

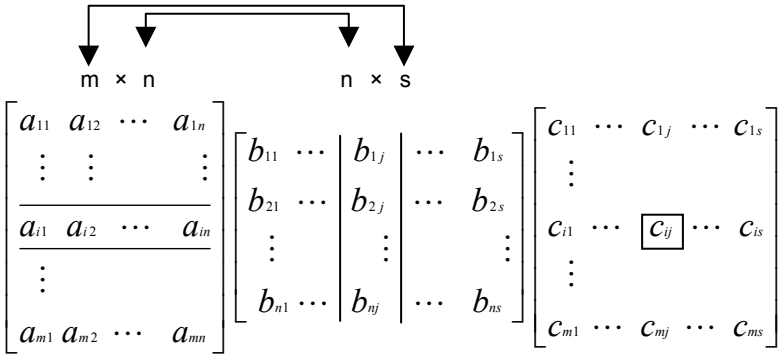
2.3 矩陣

1. 矩陣相乘

假設 A = (a_{ij})是一 m*n 的矩陣，而 B = (b_{ij})為 n*s 的矩陣，則 AB 的乘積為 m*s 的矩陣

$$(AB)_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}$$

如下圖所示：



表示 c 陣列的第 i 列第 j 行的值為 a 陣列的第 i 列乘以 b 陣列的第 j 行

即 $C_{ij} = a_{i1}*b_{1j}+a_{i2}*b_{2j}+\cdots+a_{in}*b_{nj}$

範例》

$$\begin{bmatrix} 2 & 1 & -3 \\ -2 & 2 & 4 \end{bmatrix} \begin{bmatrix} -1 & 2 \\ 0 & -3 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 2(-1)+1(0)+(-3)2 & 2(2)+1(-3)+(-3)1 \\ (-2)(-1)+2(0)+4(2) & (-2)2+2(-3)+4(1) \end{bmatrix}$$

$$= \begin{bmatrix} -8 & -2 \\ 10 & -6 \end{bmatrix}$$

矩陣相乘的片段程式如下：

JAVA 片段程式》 矩陣相乘

```
public static void matrix(int A[][], int B[][])
{
    int i=0, j=0, k=0;

    // 將 A 矩陣每一列元素與 B 矩陣每一列元素
    // 相乘之和放入 C 矩陣之中
    for ( i = 0 ; i < N ; i++ )
        for ( j = 0 ; j < N ; j++ ) {
```

```
            int sum = 0 ;
            for ( k = 0 ; k < N ; k++ )
                sum = sum + A[i][k] * B[k][j] ;
            C[i][j] = sum ;
        }
}
```

有關矩陣相乘之程式實作，請參閱 2.8 節。

2. 稀疏矩陣

若一矩陣中有大多數元素為 0 時，則稱此矩陣為稀疏矩陣(sparse matrix)，到底要多少個 0 才算是疏稀，則沒有絕對的定義，一般而言，大於 1/2 個就可稱之，如圖 2-4 是一稀疏矩陣。

$$\begin{bmatrix} 0 & 15 & 0 & 0 & -8 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 \\ 0 & 0 & 18 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 \\ 72 & 0 & 0 & 0 & 20 & 0 \end{bmatrix}$$

圖 2-4 稀疏矩陣

此矩陣共有 36 個元素，但只有 7 個元素非為 0，因此 0 的個數佔了 80%左右，若是將 1000×1000 矩陣以二維陣列來儲存，勢必會浪費許多空間，因為這些 0 根本不必管它，只要儲存非零的元素即可，因此，我們可以下列的資料結構表示之。

(i, j, value)，其中 i 表示第幾列，j 表示第幾行，而 value 表示要儲存的值，若將此矩陣的非零值儲存於二維陣列 A(0：n, 1：3) 的結構中，其中 n 表示非零的數字。上一矩陣若以此結構表示的話，情況如下所示(假設二維陣列的第一個元素為 A(1, 1))：

	1)	2)	3)
A(0,	6	6	7
A(1,	1	2	15
A(2,	1	5	–8
A(3,	3	4	–6
A(4,	4	3	18
A(5,	5	6	16
A(6,	6	1	72
A(7,	6	5	20

其中 A(0, 1)=6 表示有 6 列，A(0, 2)=6 表示有 6 行，而 A(0, 3)=7 表示有 7 個非零值。而 A(1, 1)=1，A(1, 2)=2 及 A(1, 3)=15，表示第一列，第 2 行的值為 15，餘此類推。

練習題

撰寫一 Java 程式，將圖 2-4 稀疏矩陣的非零元素，存於一個名為 sm 的二維陣列中。

2.4 多項式表示法

有一多項式 $p=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$，我們稱 p 為 n 次多項式，$a_ix^j$ 是多項式的項 ($0 \le i \le n$, $1 \le j \le n$)，其中 a_i 為係數，x 為變數，j 為指數。一般多項式可以使用線性串列來表示其資料結構，也可以使用鏈結串列來表示(在第 4 章討論)。

多項式使用線性串列來表示有兩種方法：

1. 使用一個 n+2 長度的陣列，依據指數由大至小依序儲存係數，陣列的第一個元素是此多項式最大的指數，如 $p = (n, a_n, a_{n-1}, \cdots, a_0)$。
2. 另一種只考慮多項式中非零項的係數，若有 m 項，則使用一個 2m+1 長度的陣列來儲存，分別儲存每一個非零項的指數與係數，而陣列中第一個元素是此多項式非零項的個數。

例如有一多項式 $p = 8x^5+6x^4+3x^2+12$，分別利用第 1 種和第 2 種方式來儲存，表示方式如下：

(1) p = (5, 8, 6, 0, 3, 0, 12)

(2) p = (4, 5, 8, 4, 6, 2, 3, 0, 12)

假若是一個兩變數的多項式，那如何利用線性串列來儲存呢？此時需要利用二維陣列，若 m，n 分別是變數最大的指數，則需要一個(m+1)*(n+1)的二維陣列。如多項式 $p_{xy}=8x^5+6x^4y^3+4x^2y+3xy^2+7$，則需要一個(5+1)*(3+1)=24 的二維陣列，表示的方法如右：

	y^0	y^1	y^2	y^3
x^0	7	0	0	0
x^1	0	0	3	0
x^2	0	4	0	0
x^3	0	0	0	0
x^4	0	0	0	6
x^5	8	0	0	0

將兩個多項式 A、B 相加，結果儲存於 C 多項式，其演算法如下：

1. A 指數=B 指數

   此時將 A 和 B 目前這一項的係數相加，成為 C 多項式的一項，之後將 A 和 B 兩個多項式向後移動。

2. A 指數 > B 指數

   此時將 A 目前這一項，成為 C 多項式的一項，之後將 A 向後移動，而 B 不動。

3. A 指數 < B 指數

   此時將 B 目前這一項，成為 C 多項式的一項，之後將 B 向後移動，而 A 不動。

有關多項相加的程式實作，請參閱 2.8 節。

練習題

1. 有一多項式 $p_x = 6x^7 + 8x^5 + 5x^4 + 3x^2 + 7$，請利用本節所提到的兩種方法表示之。
2. 有一多項式 $p_{xy} = 6x^5 + 3x^4y^3 + 2x^3y^2 - 8x^2y + 9x + 3$，試利用二維陣列表示之。

2.5 上三角形和下三角表示法

若一矩陣的對角線以下的元素均為零時，亦即 $a_{ij} = 0$，$i > j$，則稱此矩陣為上三角形矩陣(upper triangular matrix)。反之，若一矩陣的對角線以上的元素均為零，亦即 $a_{ij} = 0, i < j$，此矩陣稱為下三角形矩陣(lower triangular matrix)，如圖 2-5 所示：

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \qquad \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

(a) 上三角形矩陣　　(b) 下三角形矩陣

圖 2-5 上、下三角形矩陣

由上述得知，一個 n * n 個的上、下三角形矩陣共有[n(n+1)] / 2 個元素，依序對映至 D(1：[n(n+1)] / 2)。

1. **以列為主**

一個 n*n 的上三角形矩陣，以列為主的對映情形如下：

a_{11}	a_{12}	a_{13}	a_{14}	…	a_{22}	a_{23}	a_{24}	…	a_{ij}	…	a_{nn}
D(1)	D(2)	D(3)	D(4)	…	D(n+1)	D(n+2)	D(n+3)	…	D(k)	…	D([n(n+1)] /2)

∴ a_{ij} = D(k)，其中 k = n(i–1) – [i(i–1)]/2 +j

例如圖 2-5 之(a)的 a_{34} 元素對映 D(k)，而

k = 4(3–1) – [3(3–1)] / 2 + 4 = 8 –3 + 4 = 9

讀者可以這樣想：a_{34} 表示此元素在第 3 列第 4 行的位置，因此上面有二列的元素，而每列 4 個位置，共 8 個空間，由於此矩陣是上三角形矩陣，因此有些位置不放元素，所以必需減掉那些不放元素的空間，它有 1+2 = 3 ([i (i–1)] / 2, i=3)，然後再加上此元素在那一行(j)。

假使是一個 n*n 的下三角形矩陣，則對映的情形如下所示：

a_{11}	a_{21}	a_{22}	a_{31}	a_{32}	…	a_{ij}	…	a_{nn}
D(1)	D(2)	D(3)	D(4)	D(5)	…	D(k)	…	D([n(n+1)] /2)

∴ a_{ij} = D(k)，其中 k = [i(i–1)] / 2 + j

例如，圖 2-5 之(b)下三角形矩陣的 a_{32} 位於 D(k)，而

k = [3(3–1)] / 2 + 2 = 5

2. **以行為主**

一個 n*n 的上三角形矩陣，以行為主的對映情形如下：

a_{11}	a_{12}	a_{22}	a_{13}	a_{23}	a_{33}	…	a_{ij}	…	a_{nn}
D(1)	D(2)	D(3)	D(4)	D(5)	D(6)	…	D(k)	…	D([n(n+1)] /2)

∴ $a_{ij} = D(k)$，其中 $k = [j(j-1)] / 2 + i$

例如，圖 2-5 之(a)的 a_{34} 位於 D(k)，其中

$k = [4(4-1)] / 2 + 3 = 6 + 3 = 9$

而下三角形矩陣的對映情形如下：

a_{11}	A_{21}	A_{31}	A_{41}	…	a_{22}	a_{32}	…	a_{ij}	…	a_{nn}
D(1)	D(2)	D(3)	D(4)	…	D(n+1)	D(n+2)	…	D(k)	…	D([n(n+1)] /2)

∴ $a_{ij} = D(k)$，其中 $k = n(j-1) - [j(j-1)] / 2 + i$

如圖 2-5 之(b)的 a_{32} 位於 D(k)，其中

$k = 4(2-1) - [2(2-1)] / 2 + 3 = 4 - 1 + 3 = 6$

由此可知，上角形矩陣以列為主和下三角形矩陣以行為主的計算方式略同，而上三角形矩陣以行為主與下三角形矩陣以列為主的計算方式相同。(請讀者注意，此處設定陣列的起始元素位置為 1)

練習題

1. 試撰寫將 $B_{n\times n}$ 的上三角形，儲存於 B(1：n(n+1)/2)陣列的演算法。
2. 撰寫從 B 陣列取出 B(i, j)的演算法。

2.6 魔術方陣

有一 n*n 的方陣，其中 n 為奇數，請你在 n*n 的魔術方陣將 1 到 n^2 的整數填入其中，使其各列、各行及對角線之和皆相等。

做法很簡單，首先將 1 填入最上列的中間空格，然後往左上方走，規則如下：

(1) 以 1 的級數增加其值，並將此值填入空格；

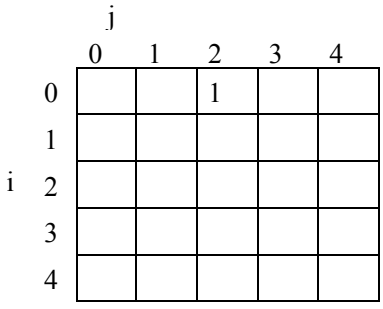
(2) 假使空格已被填滿，則在原地的下一空格填上數字，並繼續往下做；

(3) 若超出方陣，則往下到最底層或往右到最右方，視兩者中那一個有空格，將數目填上此空格；

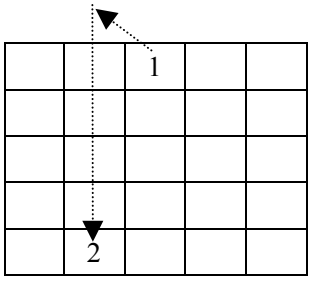
(4) 若兩者皆無空格，則在原地的下一空格填上數字。

例如，有一 5*5 的方陣，其形成魔術方陣的步驟如下，我們以上述(1)、(2)、(3)、(4)的規則來說明。

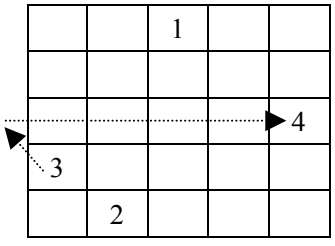
1. 將 1 填入此方陣的最上列的中間空格，如下所示：



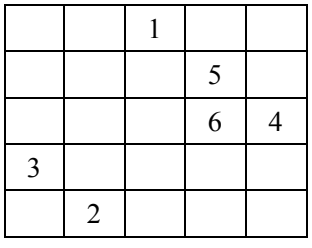
2. 承 1，往左上方走，由於超出方陣，依據規則(3)發現往下的最底層有空格，因此將 2 填上。如下所示：



3. 承 2，往左上方，依據規則(1)將 3 填上，然後再往左上方，此時，超出方陣，依據規則(3)將 4 填在最右方的空格，如下所示：



4. 承 3，往左上方，依據規則(1)將 5 填上，再往左上方時，此時方格已有數字，依據規則(2)，往 5 的下方填，如下所示：



5. 依此類推，依據上述的四個規則繼續填，填到 15 的結果如下：

15	8	1		
	14	7	5	
		13	6	4
3			12	10
9	2			11

6. 承 5，此時往左上方，發現往下的最底層和往右的最右方皆無空格，依據規則(4)，在原地的下方將此數字填上，如下所示：

15	8	1		
16	14	7	5	
		13	6	4
3			12	10
9	2			11

7. 繼續往下填，並依據規則(1)、(2)、(3)、(4)，最後的結果如下：

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

此時讀者可以算算各行、各列及對角線之和是否皆相等，其和為 65。有關奇數魔術方陣的程式實作，請參閱 2.8 節。

練習題

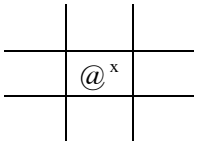
自行完成 9*9 的魔術方陣。

2.7 生命細胞遊戲

本章將以生命細胞遊戲(game of life)做為結束，此遊戲在 1970 年由英國數學家 J. H. CONWAY 所提出。生命細胞遊戲將陣列元素視為細胞，而某一細胞的鄰居乃是指在其垂直、水平、對角線相鄰之細胞(cells)。

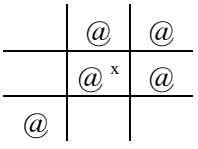
生命細胞遊戲的規則：

1. **孤單死**：若一活細胞只有一個或沒有鄰居細胞存活的，則在下一代，它將孤單而死。(下圖中以@表示一細胞，而 x 符號表示此細胞將死去)如：



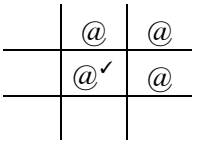
孤單死

2. **擁擠死**：一活細胞有四個或四個以上鄰居亦是活的，則在下一代，它將因擁擠而死。(下圖中以 x 符號表示此細胞將死去)如：



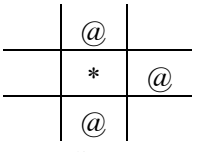
擁擠死

3. **穩定**：一活細胞有二個或三個相鄰活細胞，則下一代它將繼續生存。(下圖中以✓符號表示此細胞將繼續存活之)如：



穩定

4. **復活**：一死細胞正好有三個相鄰的活細胞，則下一代它將復活。(下圖中以*符號表示此位置會復活一細胞)如：



復活

由上規則可得：

- 某細胞若有 0 或 1 個相鄰細胞，則它在下一代將會因孤單而死。
- 某細胞若有 4, 5, 6, 7, 8 個相鄰細胞，則它在下一代將會因擁擠而死。
- 某細胞若有 2 個相鄰活細胞，則它在下一代將保持不變。
- 某細胞若有 3 個相鄰活細胞者，則不管其現在是生或死，下一代會是活的。

範例一》

		@	@			

我們將上圖填上每一細胞的相鄰活細胞個數

	0	0	0	0	0	0	
	0	1	2	2	1	0	
	0	1	@1	@1	1	0	
	0	1	2	2	1	0	
	0	0	0	0	0	0	

根據規則(1)，圖中的細胞將會孤單而死

範例二》

	0	0	0	0	0	0	
	0	1	2	2	1	0	
	0	2	@3	@3	2	0	
	0	2	@3	@3	2	0	
	0	1	2	2	1	0	
	0	0	0	0	0	0	

某一細胞若相鄰的活細胞為 2 或 3，則它將會存活下來。但圖中的死細胞其相鄰的活細胞都是小於或等於 2，故不能再生，所以已成為穩定狀態。

範例三》

	0	0	0	0	0	
	1	2	3	2	1	
	1	@1	@2	@1	1	
	1	2	3	2	1	
	0	0	0	0	0	

根據上述規則，它的下一代將為

	0	1	1	1	0	
	0	2	@1	2	0	
	0	3	@2	3	0	
	0	2	@1	2	0	
	0	1	1	1	0	

這二張圖將會來回的互換。

有關生命細胞遊戲的程式實作，請參閱 2.8 節。

2.8 程式實作

(一) 矩陣相乘

JAVA 程式語言實作》 兩個矩陣相乘

```
01  package matrix;
02
03  /**
04   *
05   * @author Bright
06   * Version 2
07   * Update date: March 18, 2017
08   */
```

```
09
10  import java.io.*;
11
12  public class Matrix
13  {
14      static int N = 3;
15      static int[][] C = new int [N][N];
16      static int sum=0;
17
18      public static void access_matrix(int A[][], int B[][])
19      {
20          int i=0, j=0, k=0;
21
22          // 將 A 矩陣每一列元素與 B 矩陣每一列元素
23          // 相乘之和放入 C 矩陣之中
24          for ( i = 0 ; i < N ; i++ )
25              for ( j = 0 ; j < N ; j++ ) {
26                  sum = 0 ;
27                  for ( k = 0 ; k < N ; k++ )
28                      sum = sum + A[i][k] * B[k][j] ;
29                  C[i][j] = sum ;
30              }
31      }
32
33      public static void output_result(int A[][], int B[][])
34      {
35          // 列出三矩陣內容
36          System.out.print("\nContent of Matrix A :\n\n");
37          output_Matrix(A) ;
38          System.out.print("\nContent of Matrix B :\n\n");
39          output_Matrix(B) ;
40          System.out.print("\nContent of Matrix C :\n\n");
41          output_Matrix(C) ;
42      }
43
44      public static void output_Matrix(int m[][])
45      {
46          int i=0, j=0 ;
47
48          for ( i = 0 ; i < N ; i++ ) {
49               for ( j = 0 ; j < N ; j++ )
50               System.out.printf("%4d", m[i][j]) ;
51               System.out.print("\n") ;
52          }
```

```
53      }
54
55      public static void main (String args[])  // 主函數
56      {
57          Matrix obj = new Matrix();
58          int[][]  A = new int[N][N];
59          int[][]  B = new int[N][N];
60
61          A[0][0]=1; A[0][1]=2; A[0][2]=3;
62          A[1][0]=4; A[1][1]=5; A[1][2]=6;
63          A[2][0]=7; A[2][1]=8; A[2][2]=9;
64
65          B[0][0]=1; B[0][1]=2; B[0][2]=3;
66          B[1][0]=1; B[1][1]=2; B[1][2]=3;
67          B[2][0]=1; B[2][1]=2; B[2][2]=3;
68
69          obj.access_matrix(A, B);
70          obj.output_result(A, B);
71      }
72  }
```

輸出結果

```
Content of Matrix A :

   1   2   3
   4   5   6
   7   8   9

Content of Matrix B :

   1   2   3
   1   2   3
   1   2   3

Content of Matrix C :

   6  12  18
  15  30  45
  24  48  72
```

(二) 多項式相加

JAVA 程式語言實作》多項式相加－利用陣列表示法做多項式相加

```
01  package polyadd;
02
03  /**
04   *
05   * @author Bright
06   * Version 2
07   * Update date: March 18, 2017
08   */
09
10  public class PolyAdd
11  {
12      static int DUMMY = -1;
13
14      public void Padd(int a[] , int b[], int c[])
15      {
16          int p = 0, q = 0, r = 0, m = 0, n = 0 ;
17          char result='\0' ;
18
19          m = a[1] ; n = b[1] ;
20          p = q = r = 2 ;
21
22          while ( (p <= 2*m) && (q <= 2*n) ) {
23              // 比較a與b的指數
24              result = compare ( a[p], b[q] ) ;
25
26              switch (result) {
27                  case '=':
28                      c[r+1] = a[p+1] + b[q+1] ; // 係數相加
29                      if ( c[r+1] != 0 ) {
30                          c[r] = a[p] ;    // 指數assign給c
31                          r += 2 ;
32                      }
33                      p += 2 ; q += 2 ;  // 移至下一個指數位置
34                      break ;
35                  case '>':
36                      c[r+1] = a[p+1] ;
37                      c[r] = a[p] ;
38                      p += 2 ; r += 2 ;
39                      break ;
40                  case '<':
```

```
                    c[r+1] = b[q+1] ;
                    c[r] = b[q] ;
                    q += 2 ; r += 2 ;
                    break ;
                }
            }
            while (p <= 2*m) { // 將多項式a 的餘項全部移至c
                c[r+1] = a[p+1] ;
                c[r] = a[p] ;
                p += 2 ; r += 2 ;
            }
            while (q <= 2*n) { // 將多項式b 的餘項全部移至c
                c[r+1] = b[q+1] ;
                c[r] = b[q] ;
                q += 2 ; r += 2 ;
            }
            c[1] = r/2 - 1 ;    // 計算c 總共有多少非零項
        }

        public char compare(int x, int y)
        {
            if (x == y)
                return '=' ;
            else if (x > y)
                return '>' ;
            else
                return '<' ;
        }

        public void output_P(int p[],int n)
        {
            int i=0 ;

            System.out.print("(") ;
            for ( i = 1 ; i <= n ; i++ )
                System.out.print(p[i] + " ") ;
            System.out.print(")") ;
        }

        public static void main (String args[])  // 主函數
        {
            // 多項式的表示方式利用只儲存非零項法
            // 分別儲存每一個非零項的指數及個數，
            // 陣列第一元素放多項式非零項個數。
```

```
85          // ex: 下列A 多項式有3 個非零項，其多項式為 :
86          //     5x 四次方 + 3x 二次方 + 2
87
88          PolyAdd obj = new PolyAdd() ;
89          int[] A = new int[8];
90          int[] B = new int[8];
91          int[] C = new int[13];  // C 的大小應為2*(A[1]+B[1]) +1
92          int index=0;
93
94          A[0]=DUMMY; A[1]=3; A[2]=4; A[3]=5; A[4]=2; A[5]=3;
95          A[6]=0; A[7]=2;
96
97          B[0]=DUMMY; B[1]=3; B[2]=3; B[3]=6; B[4]=2; B[5]=2;
98          B[6]=0; B[7]=1;
99          for (index=1;index<13;index++)
100             C[index]=0;
101
102         obj.Padd( A, B, C ) ;  // 將A 加B 放至C
103
104         // 顯示各多項式結果
105         System.out.print("\nA = ") ;
106         obj.output_P(A, A[1]*2 +1) ;  // A[1]*2 + 1 為陣列A 的大小
107         System.out.print("\nB = ") ;
108         obj.output_P(B, B[1]*2 +1) ;
109         System.out.print("\nC = ") ;
110         obj.output_P(C, C[1]*2 +1) ;
111     }
112 }
```

輸出結果

```
A = (  3  4  5  2  3  0  2  )
B = (  3  3  6  2  2  0  1  )
C = (  4  4  5  3  6  2  5  0  3  )
```

》程式解說

1. 在程式中，a[1]表示 a 多項式非零項的個數，b[1]為 b 多項式非零項的個數，一開始先將陣列的第 2 個元素指定給 p、q、r。在 p <= 2*m 及 q <= 2*n 的狀況下，才做多項式的比較動作。

2. 多項式的比較動作是比較指數，而非係數，因此在 while 敘述中，p += 2 與 q += 2 的目的，是為了取得多項式的指數。最後，while(p <= 2*m)敘述，表示當 b 的多項式已結束，則將 a 多項式的餘項搬到 c 多項式；若 while(q <= 2*n)

條件成立，表示 a 多項式已結束，則將 b 多項式的餘項搬到 c 多項式中。最後計算 c 多項式中非零項的個數。

(三) 奇數魔術方陣

JAVA 程式語言實作》奇數魔術方陣

```
01  package oddmagic;
02
03  /**
04   *
05   * @author Bright
06   * Version 2
07   * Update date: March 18, 2017
08   */
09
10  import java.io.*;
11  import java.util.Scanner;
12
13  public class OddMagic
14  {
15      static int MAX=15;
16      static int[][] Square = new int[MAX][MAX];  // 定義整數矩陣
17      static int N=0;  // 矩陣行列大小變數
18      static Scanner keyboard = new Scanner(System.in);
19
20      public static void init()
21      {
22          String st = "";
23
24          // 讀取魔術矩陣的大小N,N 為奇數且 0 <= N <= 15
25          do {
26              System.out.print("\nEnter odd matrix size : ");
27              N = keyboard.nextInt();
28              if ( N % 2 == 0 || N <= 0 || N > 15)
29                   System.out.print("Should be > 0 and < 15 odd number") ;
30              else
31                   break ;
32          } while (1 == 1);
33      }
34
35      public static void Magic()
36      {
```

```
37          int i=0, j=0, p=0, q=0, key=0 ;
38
39          // 初始化矩陣內容,矩陣全部清0
40          for (i = 0 ; i < N ; i++)
41              for (j = 0 ; j < N ; j++)
42                  Square[i][j] = 0 ;
43          Square[0][(N-1)/2] = 1 ; // 將1 放至最上列中間位置
44          key = 2 ;
45          i = 0 ; j = (N-1)/2 ;    // i,j 記錄目前所在位置
46          while ( key <= N*N ) {
47              p = (i-1) % N ;  // p, q 為下一步位置, i, j 各減1 表往西北角移動
48              q = (j-1) % N ;
49              // p < 0 (超出方陣上方) 則將p 移至N-1(最下列)
50              if (p < 0)
51                  p = N - 1 ;
52              // q < 0 (超出方陣左方) 則將q 移至N-1(最右行)
53              if (q < 0)
54                  q = N - 1 ;
55              if (Square[p][q] != 0)  // 判斷下一步是否已有數字
56                  i = (i + 1) % N ;  // 已有數字，則 i 往下填在原值下方，j 不變
57              else {
58                  i = p ;      // 將下一步位置指定給目前位置
59                  j = q ;
60              }
61              Square[i][j] = key ;
62              key++;
63          }
64      }
65
66      public static void output()
67      {
68          int i = 0, j = 0;
69
70          // 顯示魔術矩陣結果
71          System.out.print("\nThe " + N + "*" + N + " Magic Matrix\n") ;
72          System.out.print("------------------------\n") ;
73          for (i = 0 ; i < N ; i++) {
74               for (j = 0 ; j < N ; j++)
75                    System.out.printf("%4d ", Square[i][j]) ;
76               System.out.print("\n") ;
77          }
78      }
79
80      public static void main (String args[])  // 主函數
```

```
81      {
82          OddMagic obj = new OddMagic();
83          int i=0, j=0;
84
85          obj.init() ;
86          obj.Magic() ; // 將square 變為N x N 的魔術矩陣
87          obj.output() ;
88      }
89  }
```

輸出結果

```
Enter odd matrix size : 5
The 5*5 Magic Matrix
- - - - - - - - - - - - - - - - - - - - - - - - - -

15  8   1   24  17
16  14  7   5   23
22  20  13  6   4
3   21  19  12  10
9   2   25  18  11
```

另一個輸出結果

```
Enter odd matrix size : 7
The 7*7 Magic Matrix
- - - - - - - - - - - - - - - - - - - - - - - - - -

28  19  10  1   48  39  30
29  27  18  9   7   47  38
37  35  26  17  8   6   46
45  36  34  25  16  14  5
4   44  42  33  24  15  13
12  3   43  41  32  23  21
20  11  2   49  40  31  22
```

》程式解說

1. 先將方陣 Square 中的每一個元素皆設為 0，在最上列的中間方格 Square[0][(N–1)/2]填 1。接下來的 while(key <= N*N)內的敘述會不斷執行，直到方陣完全走完為止，其中(p, q)為下一步的位置，當 p < 0 表示超出方陣上方，依據規則調整 p 至最下列(N–1)。同理，當 q < 0 表示超出方陣左方，調整 q 至最右行(N–1)的位置。

2. if (Square[p][q] != 0)是判斷下一方格是否已有數字，若發現已有數字，則移動目前位置至原來的位置(i, j)下方；若下一方格沒有數字，則移動目前位置至下一步位置(p, q)，將數字填入方格中。

以上述的 5*5 方陣為例，來說明魔術方陣的演算法：

1. 首先將 1 放在 Square[0][N–1] / 2]的方格上，若 N = 5，則此方格為第 0 列、第 2 行。
2. 將目前的方格所代表的第 N 列和第 N 行存放在 i 與 j 中，此時 i = 0，j = 2。並將 2 指定給 key。
3. 當 key ≤ 5^2 時，將(i–1) % N，即(0–1)%5 = –1；(j–1) % N，即(2–1) % 5 = 1，求目前方格左上方的座標，但因(–1, 1)已超出方陣最上方，故依規則將列座標調整至最下列 N–1 位置，即 5–1 = 4。由於(4, 1) = 0，故將 4 指定給 j，然後將 key 的值放在(i, j) = (4, 1)方格上，key++。
4. 倘若 key = 6，此時的(i, j)為(1, 3)。因為 Square[0][2]已有數字，即 Square[p][q] != 0，則將(1+1) % 5 = 2，將此數字指定給 i，即此方格為(2, 3)，表示在原來的方格往下移一格。
5. 利用同樣的方法即可完成魔術方陣。

(四) 生命細胞遊戲

JAVA 程式語言實作》生命細胞遊戲

```
01  package lifegame;
02
03  /**
04   *
05   * @author Bright
06   * Version 2
07   * Update date: March 18, 2017
08   */
09
10  import java.io.*;
11  import java.util.Scanner;
12
13  class LifeGame {
14      static int MAXROW=10, MAXCOL=25;
15      static int DEAD=0, ALIVE=1;
16      static int[][] map = new int[MAXROW][MAXCOL];
17      static int[][] newmap= new int[MAXROW][MAXCOL];
18      static int Generation=0;
19      static Scanner keyboard = new Scanner(System.in);
20
21      LifeGame() {
```

```
22              Generation = 0;
23      }
24
25      public static void init()
26      {
27              int row = 0, col = 0;
28
29          // 起始 map 狀態,一開始 cells 皆會 DEAD
30          for (row = 0 ; row < MAXROW ; row++)
31              for (col = 0 ; col < MAXCOL ; col++)
32                  map[row][col] = DEAD;
33
34          System.out.print("Game of life Program \n");
35          System.out.print("Enter (x,y) where (x,y) is a living cell\n");
36          System.out.print(" 0 <= x <= " + (MAXROW-1) + " , 0 <= y <= " + (MAXCOL-1)
37                                  + "\n");
38          System.out.print("Terminate with (x,y) = ( -1,-1)\n");
39
40          // 輸入活細胞之位置,以(-1,-1)結束輸入
41          do {
42              System.out.print("x-->");
43              System.out.flush();
44              row = keyboard.nextInt();
45
46              System.out.print("y-->");
47              col = keyboard.nextInt();
48              if (0 <= row && row < MAXROW && 0 <= col && col < MAXCOL)
49                  map[row][col] = ALIVE;
50              else
51                  System.out.print("(x,y) exceeds map ranage!\n");
52          }  while (row != -1 || col != -1);
53      }
54
55      public static int Neighbors(int row,int col)
56      {
57          int count=0, c=0, r=0;
58
59          // 計算每一個 cell 的鄰居個數
60          // 因為 cell 本身亦被當做鄰居計算
61          // 故最後還要調整
62
63          for (r = row -1 ; r <= row +1; r++)
```

```
64              for (c = col -1 ;  c <= col + 1 ; c++) {
65                  if (r < 0 || r >= MAXROW || c < 0 || c >= MAXCOL)
66                      continue;
67                  if (map[r][c] == ALIVE)
68                        count++;
69              }
70          // 調整鄰居個數
71          if (map[row][col] == ALIVE)
72              count-- ;
73          return count ;
74      }
75
76      // 顯示目前細胞狀態
77      public static void output_map()
78      {
79          int row=0, col=0;
80          String space = " ";
81
82          System.out.print(space + "Game of life cell status\n") ;
83          System.out.print(space + "------Generation " + (++Generation) +
84                                   "-------\n");
85          for ( row = 0 ; row < MAXROW ; row++ ) {
86              System.out.print("\n");
87              System.out.print(space);
88              for ( col = 0 ; col < MAXCOL ; col++ )
89                  if ( map[row][col] == ALIVE )
90                      System.out.print("@");
91                  else
92                      System.out.print("-");
93          }
94      }
95
96      public static void access()
97      {
98          int row=0, col=0;
99          String ans="";
100
101         do {
102             // 計算每一個(row,col)之 cell 的鄰居個數
103             // 依此個數決定其下一代是生是死。
104             // 將下一代的 map 暫存在 newmap 以防 overwrite map。
105             for (row = 0 ; row < MAXROW ; row++)
```

```
106                 for (col = 0 ; col < MAXCOL ; col++)
107                     switch(Neighbors(row, col)) {
108                         case 0 :
109                         case 1 :
110                         case 4 :
111                         case 5 :
112                         case 6 :
113                         case 7 :
114                         case 8 :
115                             newmap[row][col] = DEAD;
116                             break;
117                         case 2 :
118                             newmap[row][col] = map[row][col];
119                             break;
120                         case 3 :
121                             newmap[row][col] = ALIVE;
122                             break;
123                     }
124             Copymap();  // 將newmap copy to map
125             do {
126                 System.out.print("\nContinue next Generation ?(y/n): ");
127                 ans = keyboard.next();
128 
129                 if (ans.equals("y") || ans.equals("n"))
130                     break;
131             } while (true);
132             if (ans.equals("y"))
133                 output_map();
134         }  while (ans.equals("y"));
135     }
136 
137     // 將newmap copy 至map 中
138     public static void Copymap()
139     {
140         int row=0, col=0;
141 
142         for (row = 0 ; row < MAXROW ; row++)
143             for (col = 0 ; col < MAXCOL ; col++)
144                 map[row][col] = newmap[row][col];
145     }
146 
147     public static void main (String args[]) //主函數
```

```
148         {
149             LifeGame obj = new LifeGame();
150 
151             obj.init();   // 起始 map
152             obj.output_map();
153             obj.access();
154         }
155     }
```

輸出結果

(略)，請自行執行程式。

我們提供給您以下幾組測試(x, y)座標的資料。

```
(a) 3   8   (b) 4   5   (c) 3   8
    3   9       4   6       3   9
    3   10      4   7       3   10
    3   11      5   5       2   10
    3   12      5   6       4   10
    -1  -1      5   7       5   10
                6   5       -1  -1
                6   6
                -1  -1
```

此處以上述(a)的資料加以測試，其執行結果如下：

```
Game of life Program
Enter (x,y) where (x,y) is a living cell
 0 <= x <= 9 , 0 <= y <= 24
Terminate with (x,y) = ( -1,-1)
x-->3
y-->8
x-->3
y-->9
x-->3
y-->10
x-->3
y-->11
x-->3
y-->12
x-->-1
y-->-1
(x,y) exceeds map ranage!
 Game of life cell status
 ------Generation 1-------

 -------------------------
 -------------------------
 -------------------------
```

```
 --------@@@@------------
 -------------------------
 -------------------------
 -------------------------
 -------------------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): y
 Game of life cell status
 ------Generation 2-------

 -------------------------
 -------------------------
 ---------@@@-------------
 ---------@@@-------------
 ---------@@@-------------
 -------------------------
 -------------------------
 -------------------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): y
 Game of life cell status
 ------Generation 3-------

 -------------------------
 ----------@--------------
 ---------@-@-------------
 --------@---@------------
 ---------@-@-------------
 ----------@--------------
 -------------------------
 -------------------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): y
 Game of life cell status
 ------Generation 4-------

 -------------------------
 ----------@--------------
 ---------@@@-------------
 --------@@-@@------------
 ---------@@@-------------
 ----------@--------------
 -------------------------
 -------------------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): y
```

```
 Game of life cell status
 ------Generation 5-------

 -------------------------
 ---------@@@-------------
 --------@---@------------
 --------@---@------------
 --------@---@------------
 ---------@@@-------------
 -------------------------
 -------------------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): y
 Game of life cell status
 ------Generation 6-------

 ----------@--------------
 ---------@@@-------------
 --------@-@-@------------
 -------@@@-@@@-----------
 --------@-@-@------------
 ---------@@@-------------
 ----------@--------------
 -------------------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): y
 Game of life cell status
 ------Generation 7-------

 ---------@@@-------------
 -------------------------
 -------@-----@-----------
 -------@-----@-----------
 -------@-----@-----------
 -------------------------
 ---------@@@-------------
 -------------------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): y
 Game of life cell status
 ------Generation 8-------

 ----------@--------------
 ----------@--------------
 -------------------------
 ------@@@---@@@----------
 -------------------------
```

```
 ----------@--------------
 ----------@--------------
 ----------@--------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): y
 Game of life cell status
 ------Generation 9-------

 -------------------------
 -------------------------
 -------@-----@-----------
 -------@-----@-----------
 -------@-----@-----------
 -------------------------
 ---------@@@-------------
 -------------------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): y
 Game of life cell status
 ------Generation 10-------

 -------------------------
 -------------------------
 -------------------------
 ------@@@---@@@----------
 -------------------------
 ----------@--------------
 ----------@--------------
 ----------@--------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): y
 Game of life cell status
 ------Generation 11-------

 -------------------------
 -------------------------
 -------@-----@-----------
 -------@-----@-----------
 -------@-----@-----------
 -------------------------
 ---------@@@-------------
 -------------------------
 -------------------------
 -------------------------
Continue next Generation?(y/n): n
```

2.9 動動腦時間

1. 假設有一陣列 A，其 A(0, 0)與 A(2, 2)的位址分別在$(1204)_8$與$(1244)_8$，求 A(3, 3)的位址(以 8 進位表示)。[2.1]

2. 有一三維陣列 A(–3：2, –2：4, 0：3)，以列為主排列，陣列的起始位址是 318，試求 A(1, 3, 2)所在的位址。[2.1]

3. 有一二維陣列 A(0：m–1, 0：n–1)，假設 A(3, 2)在 1110，而 A(2, 3)在 1115，若每個元素佔一個空間，請問 A(1, 4)所在的位址。[2.1]

4. 若將一對稱矩陣(symmetric matrix)視為上三角形矩陣來儲存，亦即 a_{11} 儲存在 A(1)，$a_{12} = a_{21}$ 儲存在 A(2)，a_{22} 在 A(3)，$a_{13} = a_{31}$ 在 A(4)，$a_{23} = a_{32}$ 在 A(5)，及 a_{ij} 在 A(k)地方。

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \qquad \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22} & a_{23} & a_{24} \\ & & a_{33} & a_{34} \\ & & & a_{44} \end{bmatrix}$$

   試求 A(i, j)儲存的位址(可用 MAX 與 MIN 函數來表示，其中 MAX 函數表示取 i, j 的最大值，MIN 函數則是取 i, j 最小值。)[2.5]

5. 有一正方形矩陣，其存放在一維陣列的形式如下：

$$\begin{bmatrix} A(1) & A(2) & A(5) & A(10) & \cdots \\ A(4) & A(3) & A(6) & A(11) & \cdots \\ A(9) & A(8) & A(7) & A(12) & \cdots \\ A(16) & A(15) & A(14) & A(13) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \cdots \end{bmatrix}$$

   讓 a_{ij} 儲存在 A(k)，試求 A(i, j)所在的位址，可用 MAX 及 MIN 函數來表示。[2.5]

6. 試回答下列問題：[2.5]

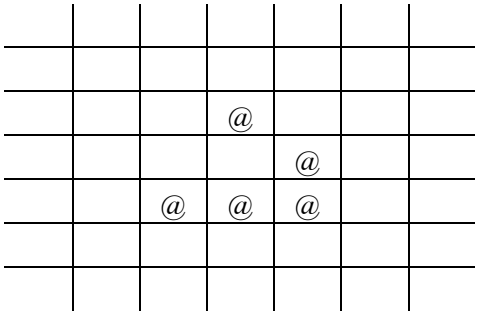
   (a) 撰寫一演算法將 A_{nxn} 的下三角形儲存於一個 B(1：n(n+1)/2)的陣列中

   (b) 撰寫一演算法從上述的陣列 B 中取出 A(i, j)

7. 在 2.6 節我們談到一個很有趣的魔術方陣，首先在第一列的中間填上 1，之後往左上方走，再遵循一些規則便可完成。如今，若改變方向，填上 1 之後，往右上方走，是否也可以完成魔術方陣呢？略述您的規則。[2.6]

8. 試完成下列生命細胞遊戲[2.7]

(a)



(b)