CH2、Array

陣列

目錄：

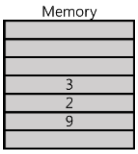
陣列

Def：Array 是用來表示 Order List(有序串列)的一種資料結構，可稱為”Dense List”或”Sequential List”

例 ：Data：3, 2, 9

3	2	9

相鄰區塊連續往下存放



特色：

1. 佔用連續的 Memory Space
2. Array 中存放”相同型別”的元素：int A[n]
3. 需事先宣告 Array 大小(彈性不佳, vs Linked List)
4. Support Sequential/Random Access(隨機存取)：速度非常快

Array 位址之計算(所在地 Loaction)

一維陣列

宣告一：A[1: n]，求 A[i]之 Location ?

	1	2	3	4	5	6	…	i	…	n-1	n
A											

*l_0 +　　　(i-1)　　　*d*

start address　　欲跳過的格數　　格子大小

宣告二：A[l: u]，求 A[i]之 Location ?

	l	l+1	l+2					i		u-1	u
A											

*l_0 +　　　(i-l)　　　*d*

start address　　欲跳過的格數　　格子大小

Note：C 語言、C++、Java，都是從 0 開始編號

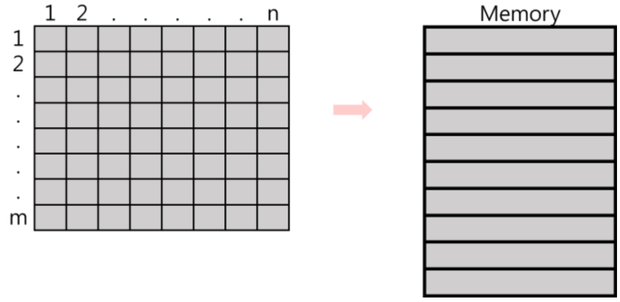
例 ：A 陣列 (-3: 9)，又 l_0=100，d=4，問 A[5]之 Location 為何 ?

*100+[5-(-3)]*4 = 132 => O(1)*

二維陣列

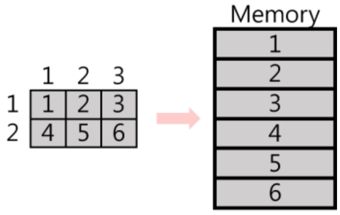
位址計算方式：

宣告一：二維陣列 m 列, n 行 => A[1: m，1: n]

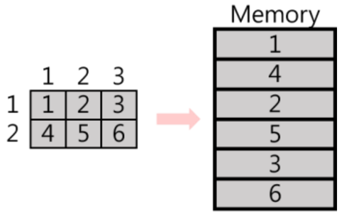


存放方式：

1. Row-Major(RM)

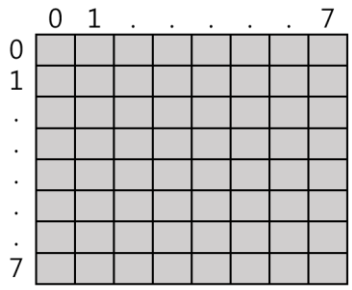


2. Column-Major(CM)



Note：C 語言、C++、Java，都是採"Row-Major"方式來處理二維陣列存放到 Memory 的順序

例 ：In C Language，則 A[8][8]圖形為何 ?

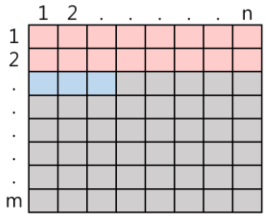


宣告一：二維陣列 m 列, n 行 => A[1: m，1: n]

1. 採 Raw-Major，求 A[i, j]位址為何？

$=l_0 + [(i-1)*n + (j-1)]$ $*d$

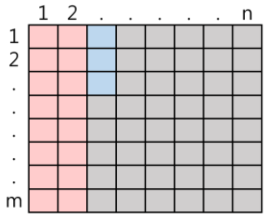
格數 *元素大小*



2. 採 Column-Major，求 A[i, j]位址為何？

$=l_0 + [(j-1)*n + (i-1)]$ $*d$

格數 *元素大小*



宣告二：二維陣列 u_1-l_1+1 列, u_2-l_2+1 行 => A[l_1: u_1, l_2: u_2]

1. 採 Raw-Major，求 A[i, j]位址為何？

$=l_0 + [(i-1_1)*(u_2-l_2+1) + (j-l_2)]$ $*d$

格數 *元素大小*

2. 採 Column-Major，求 A[i, j]位址為何？

$=l_0 + [(j-1_2)*(u_1-l_1+1) + (i-1_1)]$ $*d$

格數 *元素大小*

[考型]

1. 全部已知量皆給，求 A[i, j]之 Location？

例 1：A[-3: 8, -5: 14]，l0=100, d=2，問 A[3, 8]的 Row-Major 之 Location？

$=l_0 + [(i-l_1)*(u_2-l_2+1) + (j-l_2)] * d = 100 + [(3-(-3))*(14-(-5)+1) + (8-(-5))]*2 = 366$

例 2：A[-3: 8, -5: 14]，l0=100, d=2，問 A[3, 8]的 Column-Major 之 Location？

$=l_0 + [(j-l_2)*(u_1-l_1+1) + (i-l_1)] * d = 100 + [(8-(-5))*(8-(-3)+1) + (3-(-3))]*2 = 424$

2. 給 2 個已知量，自行判別 Row-Major 或 Column-Major，不給 l_0*(d 沒給，則 Default 帶 1 即可)*，則：

(1) Row-Major：由左往右看

$A[i_2, j_2] = A[i, j] + [(i_2-i_1)*n + (j_2-j_1)]*d$

(可得行數，但列數不得而知)

(2) Column-Major：由右往左看

$A[i_2, j_2] = A[i, j] + [(j_2-j_1) *m + (i_2-i_1)]*d$

(可得列數，但行數不得而知)

例：A[4, 2]之 address=1978, A[2, 3]之 address=1986，問 A[3,8]之 Location？

步驟：

1. *check Row-Major 還是 Column-Major*
   A[4, 2] = 1978
   v ^ ^
   A[2, 3] = 1986
   看開口，一樣者可決定何種 Major
2. *求行(Row-Major)或列數*
   *A[2, 3] = A[4, 2] + [(3-2)*n + (2-4)]*1 => 1986=1978+n-2 => n=10*
3. *求 A[i, j]*
   *A[3, 8] = A[4, 2] + [(8, 2)*10 + (3-4)]*1 = 1978 + 59 = 2037*

3. 同題型 2，但判別不出是 Row-Major 或 Column-Major
   例：A[3,3]之 address = 121，A[6,4]之 address = 159，問 A[4,9]之 Location?

步驟：

1. *check Row-Major 或 Column-Major*
   無法確認，故都要求出
2. *求行和列數*
   $A[i_2, j_2] = A[i_1, j_1] + [(i_2-i_1)*n + (j_2-j_1)] *d$ => 159 = 121 + [(6-3) * n + (4-3)] *1 => n=37/3 (不合理)
   $A[i_2, j_2] = A[i_1, j_1] + [(j_2-j_1)*m + (i_2-i_1)] *d$ => 159 = 121 + [(4-3) * n + (6-3)] *1 => n=35 (合理)
3. *求 A[i,j]*
   $A[i_3, j_3] = A[i_1, j_1] + [(j_3-j_1)*m + (i_3-i_1)] *d$ => 159 = 121 + [(9-3) * 35 + (4-3)] *1 = 332

4.
   例：已知 A[1,1]之 address=2，A[2,3]之 address=18，A[3,2]之 address=28，求 A[5,4]之 Location?

1. *check Row-Major 或 Column-Major*
   A[2,3]之 address=18，A[3,2]之 address=28 => Row-Major
2. *求行 or 列"及 d(元素)"*
   *A[2, 3] = A[1, 1] + [(2-1)*n + (3-1)] *d => 18 = 2 + nd + 2d*
   *A[3, 2] = A[1, 1] + [(33-1)*n + (2-1)] *d => 28 = 2 + 2nd + d*
   => d=2, n=6
3. *求 A[i,j]*
   *A[5, 4] = A[1, 1] + [(5-1)*6 + (4-1)] *2 =2 + 54 = 56*

三維陣列

宣告一：$A[1: u_1, 1: u_2, 1: u_3]$

1. 以 Row-Major 求 A[i, j, k]之 Location：
   $l_0 + [(i-1)*u_2*u_3 + (j-1)*u_3 + (k-1)] *d$
2. 以 Column-Major 求 A[i, j, k]之 location：
   $l_0 + [(k-1)*u_2*u_1 + (j-1)*u_1 + (i-1)] *d$

例：A[-2: 5, -3: 2, -1: 9, 3: 7]，d=2，l_0=100，採 Column-Major，求 A[1, 0, 1, 5]之 Location？

*100+ [(5-3)*11*6*8 + (1-(-1))*6*8 + (0-(-1))*8 + (1-(-2))] *2 = 2458*

n 維陣列

一樣分成 Row-Major 與 Column-Major：A[1: u_1, 1: u_2, 1: u_3, …]

1. 以 Row-Major 求 A[i_1, i_2, i_3, …, i_n]之 Location：

$$
\begin{aligned}
l_0 &+ [\ (i_1-1)*u_2*u_3\ldots *u_n \\
&+ \quad (i_2-1)*u_3*u_4\ldots*u_n \\
&+ \quad (i_3-1)*u_4*u_5\ldots*u_n \\
&\ldots \\
&+ \quad (i_n-1)]\ *d = \sum [(i_j-1)*\pi^n\ u_{j+1}]*d
\end{aligned}
$$

2. 以 Column-Major 求 A[i_1, i_2, i_3, …, i_n]之 location：

$$
\begin{aligned}
l_0 &+ [\ (i_1-1)*u_{n-1}*u_{n-2}\ldots*u_1 \\
&+ \quad (i_2-1)*u_{n-2}*u_{n-3}\ldots*u_1 \\
&+ \quad (i_3-1)*u_{n-3}*u_{n-4}\ldots*u_1 \\
&\ldots \\
&+ \quad (i_n-1)]\ *d= \sum [(i_j-1)*\pi^n\ u_{j+1}]*d
\end{aligned}
$$

多項式表式法

一、Array(連續性)：分 2 種

二、Linked List(不連續性)：亦分 2 種*(CH4 再談)*

1. 依指數由高到低，依序存放對應的係數
   令最高指數為 n => 須準備 n+2 格
   例：f(x) = 3x5+4x3+9x+2

   n=5，∴給 A[1: 7]

A	*5*	*3*	*0*	*4*	*0*	*9*	*2*

   Note：若 0 項次很多，則不適用：f(x)=3^{100} + 9

2. 只存非 0 項元素的指數、係數
   若一多項式有 n 個非 0 項 => 須準備 A[1: 2n+1]格
   例：f(x)=3^{100} + 9

   A[1: 5]的一維陣列

A	*2*	*3*	*100*	*9*	*0*

   或

A	*2*	*100*	*3*	*0*	*9*

   Note：當非 0 項很多，則不適用

特殊矩陣

1. 稀疏
2. 上、下三角
3. 對稱
4. 寬帶

Sparse Matrix(稀疏矩陣)

Def：指非 0 項元素很少的矩陣

例：矩陣 $A_{4\times3}$

$$\begin{array}{c|ccc|} & 1 & 2 & 3 \\ 1 & 0 & 7 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & -5 & 0 & 0 \\ 4 & 0 & 0 & 6 \end{array}$$

3-Tuple(較省空間的方式)

Def：準備一二維陣列 A[0: k, 1: 3]，其中 k 為非 0 項

承上例：k=3，因此 A[0: 3, 1: 3]

	1	2	3	
0	*4*	*3*	*3*	*列數、行數、k 值*
1	*1*	*2*	*7*	*所在列、所在行、數值*
2	*3*	*1*	*-5*	
3	*4*	*3*	*6*	

上、下三角矩陣

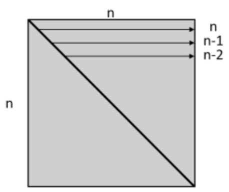
Def：

1. 上三角：即對角線以下(不含對角線)元素均為 0，$a_{ij}=0, i>j$
2. 下三角：即對角線以上(不含對角線)元素均為 0，$a_{ij}=0, i<j$

分析：

1. 以二維陣列存放 => 耗 Space

   說明：*(以上三角為例)*

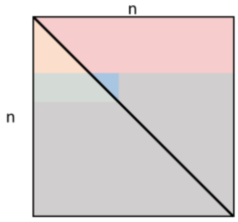
   總元素：n^2

   有用的元素(最多)：n(n+1)/2

   浪費了：$n^2-n(n+1)/2$

2. 上三角對應到一個一維陣列 B(i: n(n+1)/2)
=> 此時 a_{ij} 存到 B(k) => 求 k ?

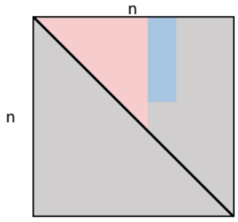
(若採 Row-Major 對應)



則 a_{ij}：$n(i-1) - i(i-1)/2 + j = k$，存到 B(k)

(若採 Column-Major 對應)

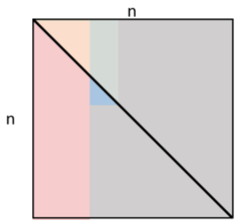
則 a_{ij}：$j(j-1)/2 + i = k$，存到 B(k)



3. 下三角對應到一個一維陣列 B(i…n(n+1)/2
=> 此時 Aij 存到 B(k) => 求 k

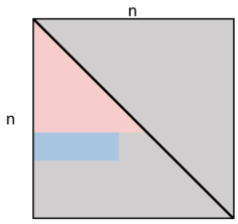
(若採 Column-Major 對應)

則 a_{ij}：$n(j-1) - j(j-1)/2 + i = k$，存到 B(k)



(若採 Row-Major 對應)

則 a_{ij}：$i(i-1)/2 + j = k$，存到 B(k)



例：A 為一 100*100 的下三角矩陣，以 Row-Major 方式存入 B[1: size]中，則：

1. size=?
2. A[70, 60]之元素會存在於 B(k), k=?
3. A[i, j]存於 B(150),問 ij=?

1. *100(100+1)/2 = 5050*
2. *70(70-1)/2 + 60 = (4900-70)/2+60= 2475*
3. *16*15/2=120, 17*16/2=136, 18*17/2=153 => i=17, j=150-136 =14*

對稱矩陣(Symmetric Matrix)

Def: 於 $A_{n\times n}$ 矩陣，若 $A_{ij}=A_{ji}$ 謂之

例：

$$\begin{vmatrix} 1 & 2 & 4 & 7 \\ 2 & 3 & 5 & 8 \\ 4 & 5 & 6 & 9 \\ 7 & 8 & 9 & 0 \end{vmatrix}$$

⇨ 較省 space 的存放方式，只存上或下三角即可

1. 上三角(Column-Major)
   $a^{ij} = j(j-1)/2 + i = k$，存到 B(k)
2. 下三角(Row-Major)：
   $a^{ij} = i(i-1)/2 + j = k$，存到 B(k)
3. 單一公式：
   $a_{ij} = \max(i, j)*(\max(i, j)-1)/2 + \min(i, j) = k \Rightarrow a^{ij}$ 存於 B(k)中

寬帶矩陣(Band Matrix)

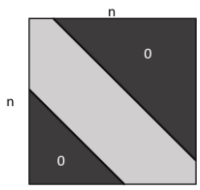
Def：$A_{n, a, b}$ 是 Band Matrix 表示一個 n×n 的矩陣中

1. 對角線(含)左下 a 條斜線是元素
2. 對角線(含)右上 b 條斜線是元素
3. 其餘為 0

$$\begin{vmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 5 & 0 \\ 6 & 7 & 8 & 9 \\ 0 & 10 & 11 & 12 \end{vmatrix}$$

例 1：$A_{n,2,2}$ 為 Band Matrix 以 Row-Major 存放，放入 B[1: size]之中，問：

1. A[i,j]=0 if(條件為何？)
2. size=?
3. A[i, j]元素存在 B(k)，公式？



1. *|i-j| > 1*
2. *3n-2(每排都為 3 個，唯最前與最末排少一元素)*
3. *(i-1)*3 -1 + (j-i+2) = k*

例 2：A4,3,2 之 Band Matrix

$$\begin{vmatrix} 1 & -1 & 0 & 0 \\ 2 & 4 & -4 & 0 \\ -3 & 8 & 7 & 2 \\ 0 & 6 & 5 & 9 \end{vmatrix}$$

=> 存入

-3	6	2	8	5	1	4	7	9	-1	-4	2

例 2：含有 $A_{100,20,30}$ 之 Band Matrix 存放跟上述相同，存到 B[1: size]，問：

1. size=?
2. A[60,65]存在 B(k)，k=?
3. A[i,j]存入 B(150)，i, j=?

1. 100+99+98+...+81 +100+99+...+71 -100 = 4275
A 排元素 B 排元素 重覆之對角線
2. 100+99+98+...+81 + 99+98+97+96 +60 = 2260
A 排元素 B 排元素 第 65 排從上到下
3. 150-81=69，i=19+69-1=87，j=69 => A[87, 69]
先扣第一排 第二排，再扣掉自身 直接算