

第六講

陣列

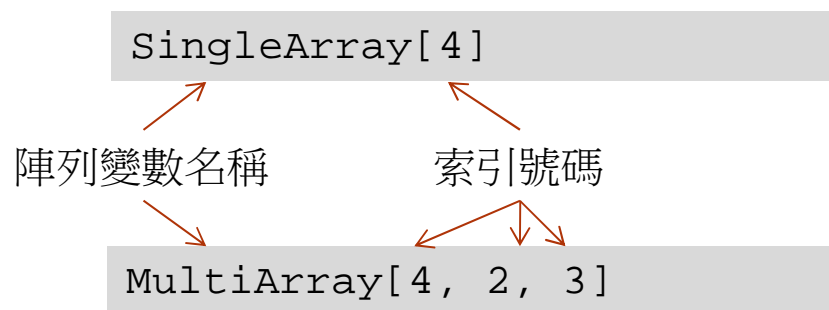
6.1

陣列簡介

陣列

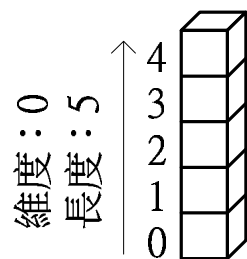
陣列 (Array) 是一組具有相同型別 (type) 的資料元素 (data element)，並且由一個單一的變數名稱來代表此一陣列。

單一元素可藉由變數名稱和介於中括弧 ([]) 內的一個 (或多個) 索引 (index) 進行存取。

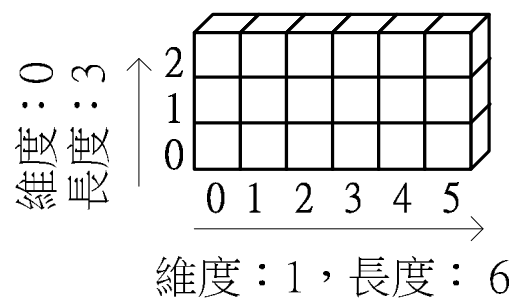


陣列

- 階層(rank) / 維度(dimension)：陣列可以有任意數量的維度。陣列的維度數目稱為它的階層。
- 維度長度(dimension length)：陣列中的每個維度都有其長度。
- 陣列長度(array length)：陣列中所有維度的所有元素。
- 陣列索引值從0開始。也就是，如果維度的長度是 n ，索引值的範圍則從0到 $n-1$ 。



一維陣列，int [5]
---Rank = 1
---Length of Array = 5



二維陣列，int [3,6]
---Rank = 2
---Length of Array = 18

陣列類型

C# 提供兩種類型陣列：

- 一維(one dimensional)陣列：可以想成是一個含許多元素的向量(vector)。
- 多維(multidimensional)陣列：可以想成在每個向量的位置也是一個陣列，稱為子陣列(subarray)。子陣列向量的每個位置可以又是一個子陣列。多維陣列又可有兩種類型：
 - 矩形(rectangular)陣列：在特定的維度下，所有子陣列的長度皆相同。使用單一對(single set)的中括弧。

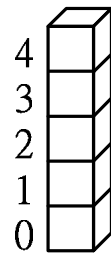
```
int x = varRectArray[3, 6, 2]
```

- 不規則(jagged)陣列：每個子陣列是獨立的陣列，可以有不同的長度。每一個維度會分別使用一對中括弧。

```
int y = varJagArray[3][6][2]
```

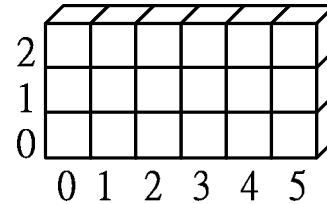
陣列型別

一維陣列



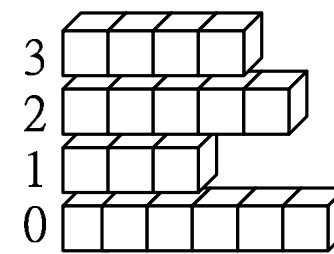
一維
`int [5]`

矩形陣列

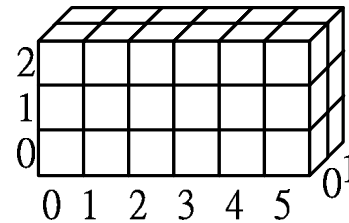


二維
`int [3,6]`

不規則陣列



不規則陣列
`int [4][]`



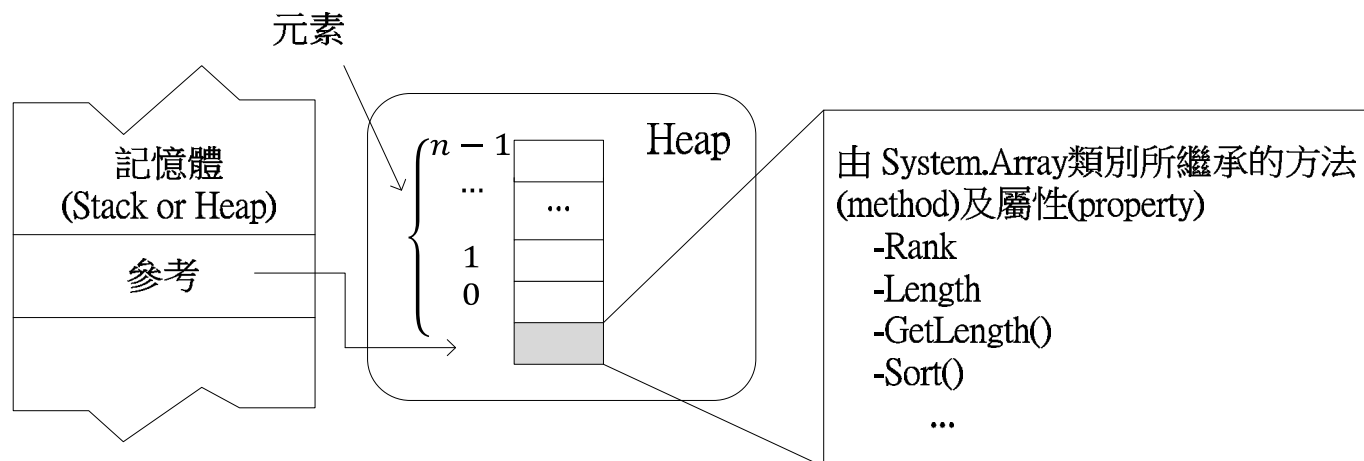
三維
`int [3,6,2]`

陣列也是一種物件

一個陣列實體(array instance)是由System.Array 類別所衍生出的物件(object)。陣列繼承了BCL一些有用的成員(member)。例如：

- Rank：該屬性會回傳陣列的維度數(number of dimensions)。
- Length：該屬性會回傳該陣列的長度(元素總數)。

陣列是參考型別(reference types)，參考(reference)可以在stack或heap中，而資料物件(data object)永遠存放在heap中。



6.2

一維陣列和矩形陣列

宣告一維陣列和矩形陣列

階層指定子(rank specifier)是介於中括弧的逗號(,)。因此，rank的數目就等於階層指定子(逗點)的總數加一。

Rank = 1

```
int[ ] myArray;
```

陣列型別

```
int[,] intArray;           // 陣列型別:2維度的int
long[, ,] longArray;       // 陣列型別:3維度的long
double[, ,] doubleArray;   // 陣列型別:3維度的double
```

當陣列被宣告時，維度的個數就固定了。然而，每個維度的大小，要到陣列被實例化(instantiated)時才決定。

```
int[3,2,6] myArray;           // 編譯錯誤
```

維度長度不允許!!

實例化一維陣列或矩形陣列

要實例化(instantiating)一個陣列，必須使用陣列創建運算式(array-creation expression)。陣列創建運算式包含 **new 運算子**，**基本型別**，及一對中括弧([])。每個維度的長度放在以逗號分隔的中括弧內。

當陣列被創建後，每個元素將會被自動初始化(initialized)為該類型的預設值。整數型別的預設值是0，浮點數型別的預設值為0.0，Boolean型別的預設值為false、參考型別的預設值為null。

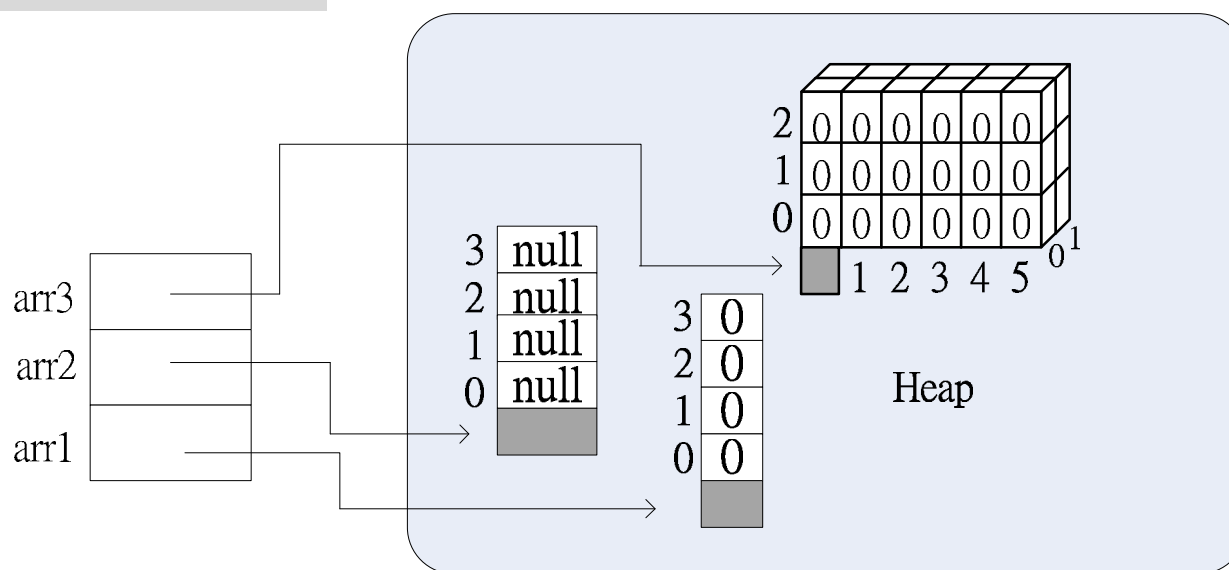
實例化一維陣列或矩形陣列

以下為一維陣列實例化範例：

```
int[] arr1 = new int[4] ;  
MyClass[] arr2 = new MyClass[4] ;
```

以下為矩形陣列實例化範例：

```
int[, ,] arr3 = new int[3,6,2] ;
```



存取矩陣元素

陣列的元素可以用一個整數的索引值去存取。

- 每一個維度的陣列索引值從0開始。

```
int[] intArr1 = new int[15];           // 宣告15個元素的一維陣列
intArr1[2] = 10;                       // 對矩陣的第三個元素寫入
int var1 = intArr1[2];                 // 讀取第三個元素

int[,] intArr2 = new int[5,10];        // 宣告二維陣列
intArr2[2,3] = 7;                     // 寫入陣列
int var2 = intArr2[2,3];              // 讀取陣列
```

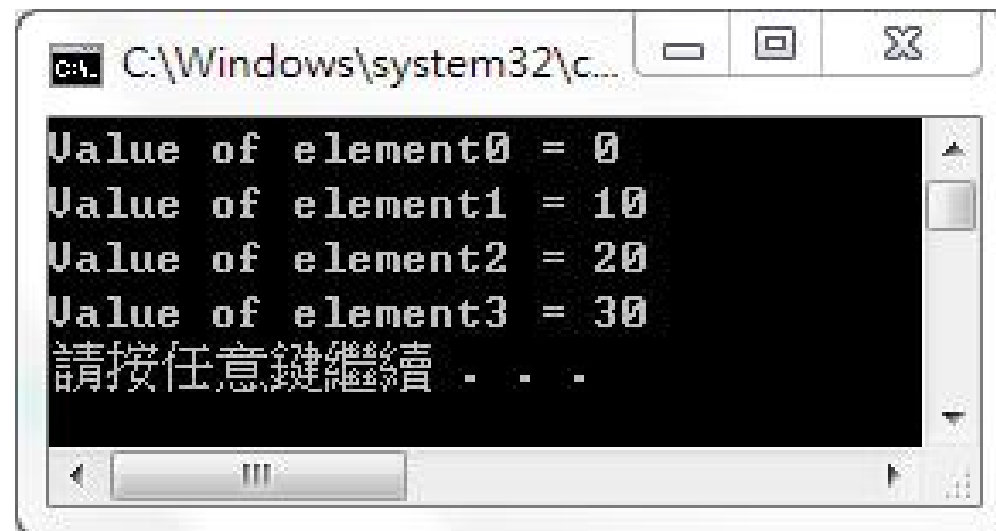
存取矩陣元素

```
int[] IntArray;           // 宣告陣列

IntArray = new int[4];

for (int i = 0; i < 4; i++) // 設定參數
    IntArray[i] = i * 10;

for (int i = 0; i < 4; i++) // 讀取和顯示每一個數值
    Console.WriteLine("Value of element{0} = {1}", i, IntArray[i]);
```



存取矩陣元素

```
for (int i = 0; i < 4; i++)    // 設定參數  
    IntArray[i] = i * 10;
```

```
for (int i = 0; i < IntArray.Length; i++)  
    IntArray[i] = i * 10;
```

```
for (int i = 0; i < IntArray.GetLength(0); i++)  
    IntArray[i] = i * 10;
```

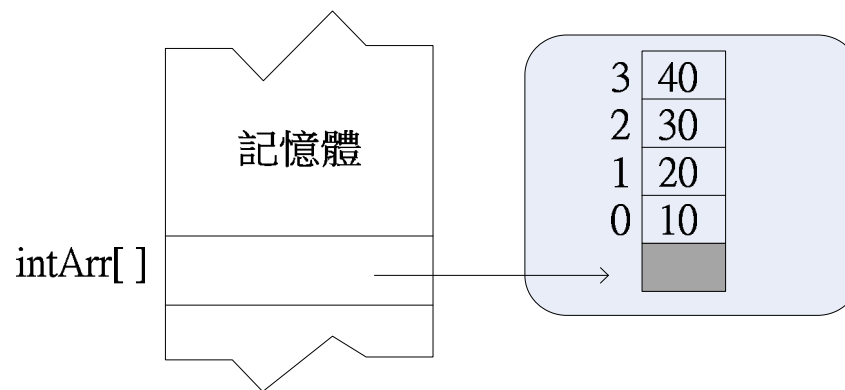
初始化陣列

對於一維陣列，我們可以在陣列創建運算式(`new`)後，立刻以初始清單(`initialization list`)的方式設定陣列元素的初始值

- 初始值須以逗號(`,`)分隔，並包含於大括弧內(`{ }`)。
- 維度長度(`dimension length`)可以省略(建議省略)

```
int[] intArr = new int[4] { 10, 20, 30, 40 };  
int[] intArr = new int[] { 10, 20, 30, 40 };  
int[] intArr = { 10, 20, 30, 40 };
```

建議用法

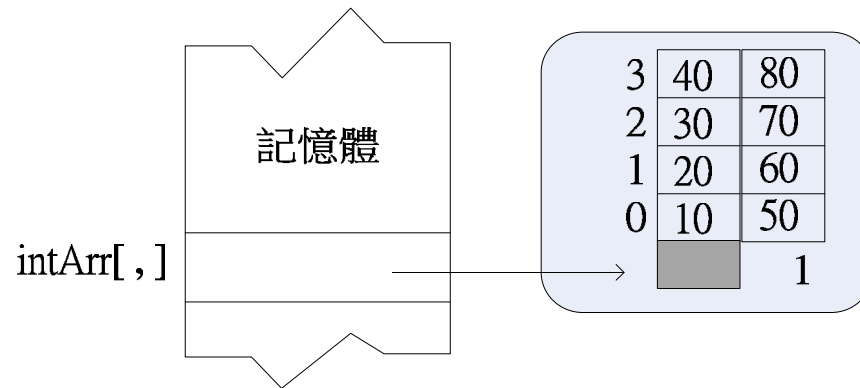


初始化陣列

對於多維矩形陣列

- 每一個初始值的向量(vector)，須以逗號(,)分隔，並包含於大括弧內({ })。
- 每一個維度也必須被巢狀(nested)包含於大括弧內({ })。

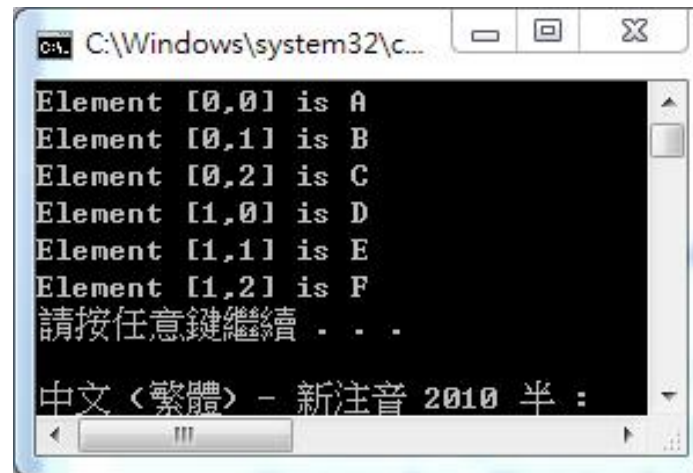
```
int[, ] intArr = new int[, ]{{10,50},{20,60},{30,70}, {40,80}};
```



```
int[, , ] intArray = new int[4,3,2]
{
    {{8,6},{5,2},{12,9}},
    {{6,4},{13,9},{18,4}},
    {{7,2},{1,13},{9,3}},
    {{4,6},{3,2},{23,8}}
};
```


初始化陣列

```
string[, ] arr = {{"A", "B", "C"}, {"D", "E", "F"}};  
  
for (int i = 0; i < arr.GetLength(0); i++)  
    for (int j = 0; j < arr.GetLength(1); j++)  
        Console.WriteLine("Element [{0},{1}] is {2}", i, j, arr[i, j]);
```



隨堂練習1

1. 試發展一個C#程式，使用者可以輸入10個整數，輸入介面如下：

Enter the number 1:

Enter the number 2:

...

Enter the number 10:

輸入同時將此10個整數依序以陣列方式儲存，並找出這10的整數的最大值，及此一數字的號碼（第一個數字的號碼為1，最後一個數字的號碼為10）。

Note：目前不考慮數字重複的情況

Console介面

輸入與出介面

輸入

```
Enter the number 1: 2  
Enter the number 2: 4  
Enter the number 3: 6  
Enter the number 4: 7  
Enter the number 5: 6  
Enter the number 6: 4  
Enter the number 7: 3  
Enter the number 8: 4  
Enter the number 9: 5  
Enter the number 10: 2
```

輸出

```
Largest number is: 7  
It's element number(s) is: 4  
請按任意鍵繼續 . . .
```

程式設計理念

```
int largest = number[0];  
int largest_count=0;  
  
for (int i = 1; i < 10; i++)  
{  
    if (number[i] > largest)  
    {  
        largest = number[i];  
        largest_count = i;  
    }  
}
```

完整程式碼

```
static void Main(string[] args)
{
    int [] number = new int[11];
    for(int i=0;i<10;i++)
    {
        Console.WriteLine("Enter the number " + i++ + ":");
        int n = Convert.ToInt32(Console.ReadLine());
        number[i--] = n;
    }
    int largest = number[0];
    for (int i = 1; i < 10; i++)
    {
        if (number[i] > largest)
        {
            largest = number[i];
        }
    }
    Console.WriteLine("最大值是: {0}", largest);
    Console.ReadKey();
}
```

隨堂練習2

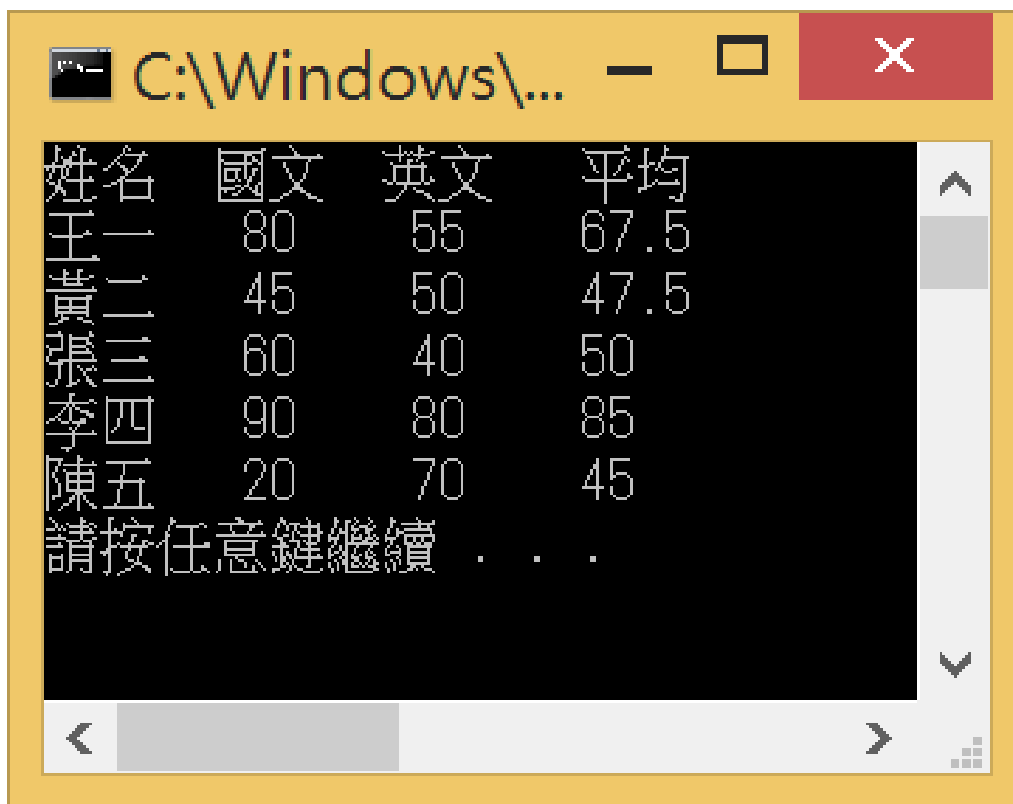
— 假設五位同學的姓名、國文成績、英文成績如下

姓名	國文成績	英文成績
王一	80	55
黃二	45	50
張三	60	40
李四	90	80
陳五	20	70

為了計算方便，上述的資料將儲存於一個Student的二維string陣列中，如下所示

```
string[,] Student = { {"王一", "80", "55"}, {"黃二", "45", "50"}, {"張三", "60", "40"}, {"李四", "90", "80"}, {"陳五", " 20 ", " 70 " } };
```

試發展一個C#程式，利用上述的Student二維陣列列
印出同學姓名、國文成績、英文成績、及平均成績
，其中 平均成績 = (國文成績+英文成績)/2



The screenshot shows a Windows-style application window with a yellow title bar. The title bar contains a small icon, the text 'C:\Windows\...', and standard minimize, maximize, and close buttons. The main content area has a black background with white text. It displays a table with four columns: '姓名' (Name), '國文' (Chinese), '英文' (English), and '平均' (Average). There are five rows of student data. Below the table, it says '請按任意鍵繼續 . . .' (Press any key to continue...). A vertical scrollbar is on the right side of the text area, and a horizontal scrollbar is at the bottom.

姓名	國文	英文	平均
王一	80	55	67.5
黃二	45	50	47.5
張三	60	40	50
李四	90	80	85
陳五	20	70	45

請按任意鍵繼續 . . .

完整程式碼

```
static void Main(string[] args)
{
    string[,] Student = { { "王一", "80", "55", "" }, { "黃二", "45", "50", "" }, { "張三", "60", "40", "" }, { "李四", "90", "80", "" }, { "陳五", "20", "70", "" } };

    Console.WriteLine("姓名" + " 國文" + " 英文" + " 平均");
    for (int i=0;i< 4; i++)
    {
        double ch = Convert.ToDouble(Student[i, 1]);
        double en = Convert.ToDouble(Student[i, 2]);
        string avg = Convert.ToString((ch + en) / 2);
        Console.WriteLine(Student[i,0]+" "+Student[i,1] + " " + Student[i,2] + " " + avg);
    }
    Console.ReadKey();
}
```


6.3

foreach 陳述式

foreach 陳述式

foreach陳述式允許你依序存取每個陣列裡的元素，語法如下：

```
foreach( Type Identifier in ArrayName)  
    Statement
```

- ArrayName是陣列名稱。
- Type型別必須與ArrayName陣列裡元素的型別一樣。
- Identifier為遞迴變數(iteration variable)的名稱，它是一個暫時性的變數，依序代表陣列中的每個元素。

foreach陳述式的運作方式如下：

1. 將ArrayName陣列第一個元素的值指定給遞迴變數。
2. 執行Statement陳述式。陳述式可以使用遞迴變數的內容
3. 完成Statement陳述式後，foreach陳述式會將陣列的下一個元素的值指定給遞迴變數，並重複2的步驟。直到所有元素都被使用過。

foreach 陳述式

```
int[] MyArr= {100, 200, 300, 400};  
  
foreach(int item in MyArr)  
    Console.WriteLine("Item: {0}", item);
```

```
Item: 100  
Item: 200  
Item: 300  
Item: 400  
請按任意鍵繼續 . . .
```

```
int[] MyArr= new int[] {100, 200, 300, 400};  
  
foreach(int item in MyArr)  
    item++; //編譯錯誤：不得改變item的值
```

foreach 陳述式用於多維陣列

矩形陣列

```
int sum =0;
int[,] MyArr = {{7,8},{9,10},{11,12}} ;

foreach(int element in MyArr)
{
    sum += element;
    Console.WriteLine("Element:{0}, Current Total: {1}", element, sum);
}
```

```
Element:7, Current Total: 7
Element:8, Current Total: 15
Element:9, Current Total: 24
Element:10, Current Total: 34
Element:11, Current Total: 45
Element:12, Current Total: 57
請按任意鍵繼續 . . .
```

6.4

常用的陣列成員

常用的陣列成員

C#陣列是由 `System.Array` class繼承來的。因此，陣列也繼承了一些有用的屬性(property)及方法(method)。

項目	類型	存在方式	含意
Rank	屬性	實例	獲取陣列的維度數
Length	屬性	實例	獲取陣列所有維度的元素總數
GetLength	方法	實例	<code>GetLength(0)</code> 取長度, <code>GetLength(1)</code> 取高度
Clear	方法	靜態	將一段元素的值設為 0 或null(空值)
Sort	方法	靜態	將一維陣列中的元素進行排序
Clone	方法	實例	執行陣列的拷貝(shallow copy)
IndexOf	方法	靜態	回傳第一個有特定值的元素索引代號
Reverse	方法	靜態	將一段元素的次序顛倒
GetUpperBound	方法	實例	獲取某一維度的元素最大索引代號

常用的陣列成員(1/2)

```
using System;
```

```
int[] MyArr = new int[] { 30, 20, 5, 75, 10 };  
//MyArr.Rank取得陣列維度,MyArr.Length取得陣列元素長度  
Console.WriteLine("rank = {0}, Length = {1}", MyArr.Rank, MyArr.Length);  
//MyArr.GetLength(0)取得陣列元素長度  
Console.WriteLine("GetLength(0) = {0}", MyArr.GetLength(0)); //實例方式呼叫  
  
foreach (int x in MyArr)  
    Console.Write("{0} ", x);  
Console.WriteLine("");  
  
Array.Sort(MyArr); //靜態方式呼叫  
foreach (int x in MyArr)  
    Console.Write("{0} ", x);  
Console.WriteLine();  
  
Array.Reverse(MyArr); //靜態方式呼叫  
foreach (int x in MyArr)  
    Console.Write("{0} ", x);  
Console.WriteLine("");
```

```
rank = 1, Length = 5  
GetLength(0) = 5  
30 20 5 75 10  
5 10 20 30 75  
75 30 20 10 5  
請按任意鍵繼續 . . .
```

常用的陣列成員(2/2)

取陣列(Array)元素長度的範例

```
//二維範例
using System;
class Hello
{
    static void Main()
    {
        int[,] m = {{7,8},{9,10},{11,12}};
        for(int i=0; i<m.GetLength(0);i++)
        {
            for(int j=0; j<m.GetLength(1);j++)
                Console.WriteLine("Element [{0},{1}] is {2}", i, j, m[i, j]);
        }
    }
}
```

```
Element [0,0] is 7
Element [0,1] is 8
Element [1,0] is 9
Element [1,1] is 10
Element [2,0] is 11
Element [2,1] is 12
```


常用的陣列成員

```
using System;
```

```
string[] MyArr = new string[] { "Cherry", "Banana", "Apple", "Watermelon",  
"Lemon" };
```

```
foreach (string x in MyArr)  
    Console.Write("{0} ", x);  
Console.WriteLine("");
```

```
int y = Array.IndexOf(MyArr, "Apple");  
Console.WriteLine("{0}", y);
```

```
Array.Clear(MyArr, 0, 3);  
foreach (string x in MyArr)  
    Console.Write("{0} ", x);  
Console.WriteLine("");
```

```
Console.WriteLine("GetUpperBound(0) = {0}", MyArr.GetUpperBound(0));
```

```
Cherry Banana Apple Watermelon Lemon  
2  
    Watermelon Lemon  
GetUpperBound(0) = 4  
請按任意鍵繼續 . . .
```

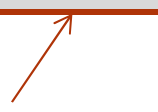
常用的陣列成員

Clone method(複製方法)會執行淺拷貝(shallow copy)，也就是只會將陣列本身複製一份。被複製的陣列會因為其型別的不同(value type 或 reference type)而有不同的結果：


- 對數值型別(value type)的陣列進行Clone，則產生兩個獨立陣列，各別有各別的元素值。
- 對參考型別(reference type)的陣列進行Clone，則產生兩個陣列，但都指向相同的元素物件。

Clone method會回傳一個物件參考，因此需要在回傳物件前指定(cast)陣列的型別。

```
int[] MyArr1 = { 10, 20, 30 };  
int[] MyArr2 = ( int[] ) MyArr1.Clone();
```



陣列型別

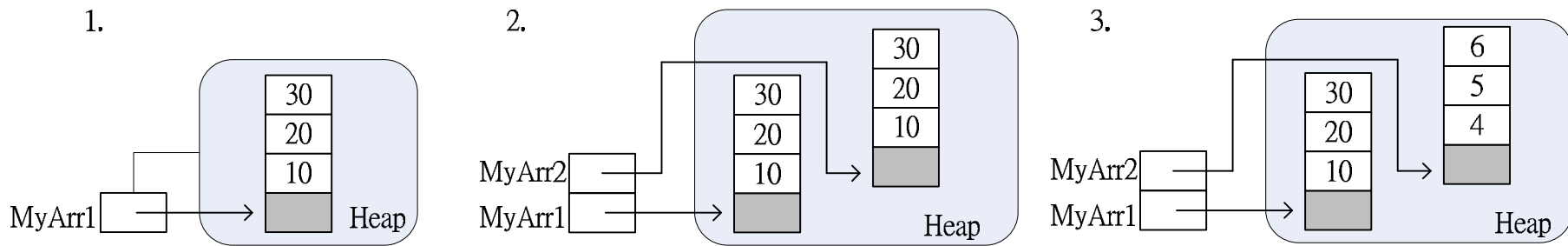


回傳一個物件

常用的陣列成員

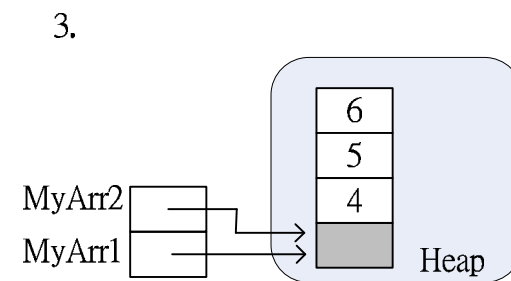
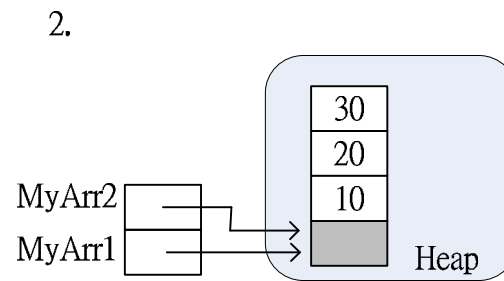
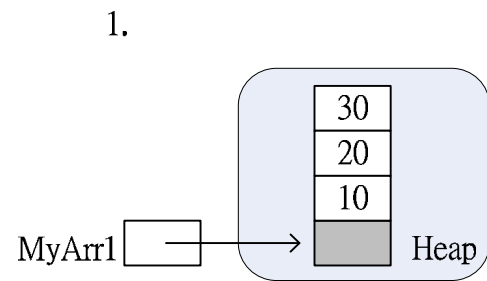
對數值型別 (value type) 的陣列進行 Clone

```
int[] MyArr1 = {10, 20, 30};  
int[] MyArr2 = (int[]) MyArr1.Clone();  
MyArr2[0] = 4;  
MyArr2[1] = 5;  
MyArr2[2] = 6;
```



常用的陣列成員

```
int[] MyArr1 = { 10, 20, 30 };  
int[] MyArr2 = MyArr1;  
MyArr2[0] = 4;  
MyArr2[1] = 5;  
MyArr2[2] = 6;
```



6.5

不規則陣列

不規則陣列

C#有提供不規則陣列(二維)的寫法，也就是陣列中可以存放陣列，每一列的長度可以不同，也稱為非矩形陣列。

宣告方法：

```
資料型別[ ][ ]陣列名稱 = new 資料型別[陣列大小][ ]
```

1. 宣告一個不規則陣列，此不規則陣列含有3列

```
int[ ][ ] jag=new int[3][ ];
```

2. 設定每個陣列大小

```
jag[0]= new int[3]; //第一列陣列大小為3
```

```
jag[1]= new int[2]; //第二列陣列大小為2
```

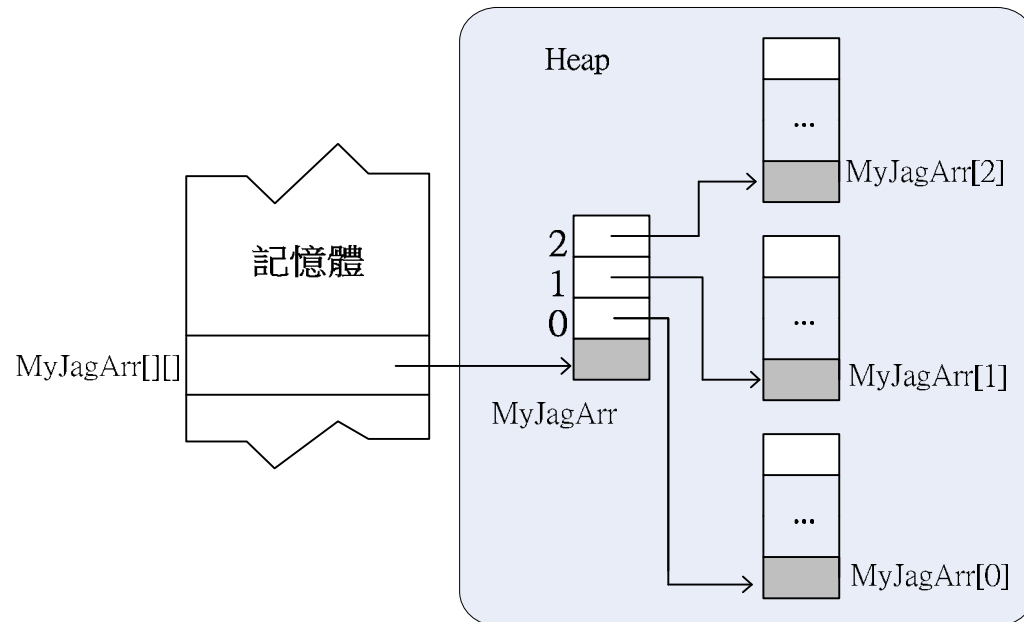
```
jag[2]= new int[4]; //第三列陣列大小為4
```

jag[0][0]	jag[0][1]	jag[0][2]	
jag[1][0]	jag[1][1]		
jag[2][0]	jag[2][1]	jag[2][2]	jag[2][3]

不規則陣列

不規則陣列 (jagged array) 與矩形陣列 (rectangular array) 不同之處在於，不規則陣列的子陣列 (subarrays) 可以有不同數目的元素。以下為宣告一個第一維度長度為3的二維不規則陣列。

```
int[][] MyJagArr = new int[3][];    // 宣告和創建頂層陣列  
...                                // 宣告和創建子陣列
```



宣告不規則陣列

不規則陣列的宣告方式：

- 一組中括弧([])代表一個維度。因此在陣列變數的宣告中，中括弧的組數的數量即為陣列的rank。
- 如同矩形陣列一樣，不規則陣列的維度長度不能在陣列型別宣告時指定。

```
int[][] Arr1;           // rank=2  
int[][][] Arr2;         // rank=3
```

我們可以將不規則陣列的宣告，與第一層陣列(the first-level array)的創見結合在一起。

```
int[ ][ ] Arr1 = new int[3][ ];
```

然而，不能將第一層陣列以外的創見也加入宣告敘述

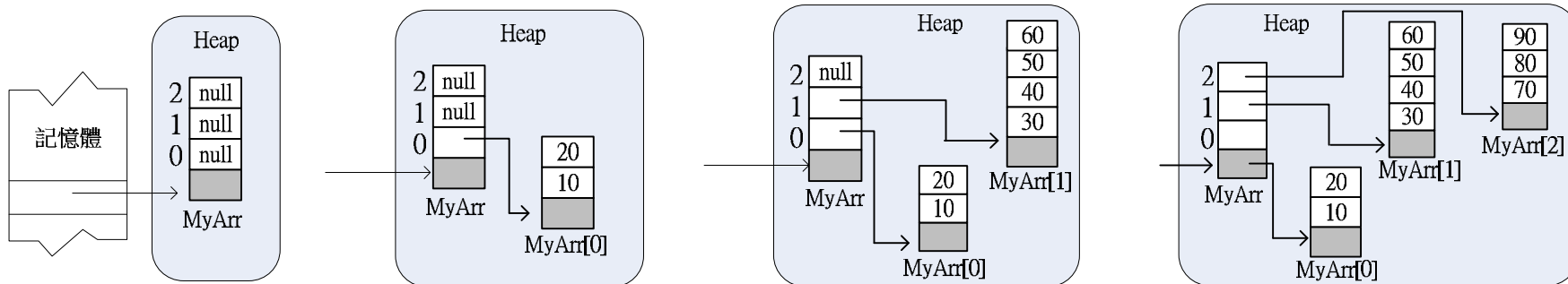
```
int[ ][ ] Arr1 = new int[3][4];           //編譯錯誤
```


實例化不規則陣列

由於不規則陣列是一個由許多獨立陣列所組成的陣列，因此每個陣列必須分別創建。實例化(instantiate)不規則陣列的步驟為：

- 將最上層(top-level)的陣列實體化
- 分別對每一個子陣列實例化，並將新創見陣列的參考(reference)指定給陣列的元素(系統會自動完成)

```
int[ ][ ] MyArr = new int[3][ ];  
MyArr[0] = new int[ ] {10, 20};  
MyArr[1] = new int[ ] {30, 40, 50, 60};  
MyArr[2] = new int[ ] {70, 80, 90};
```



不規則陣列的子陣列

由於不規則陣列中的子陣列本身就是陣列，因此在不規則陣列內可有矩形陣列的存在。GetLength(int n)方法可以得到陣列某一個維度(編號n)的長度。

```
int[,] myarr;  
myarr = new int[3][,];  
myarr[0] = new int[,] {{5,6},  
                       {7,8} };  
myarr[1] = new int[,] {{10,20,30},  
                       {40,50,60}};  
myarr[2] = new int[,] {{100,200,300,400},  
                       {500,600,700,800}};
```

```
for (int i = 0; i < myarr.GetLength(0); i++)  
{  
    for (int j = 0; j < myarr[i].GetLength(0); j++)  
    {  
        for (int k = 0; k < myarr[i].GetLength(1); k++)  
        {  
            Console.WriteLine("[{0}][{1},{2}] = {3}", i, j, k, myarr[i][j, k]);  
        }  
        Console.WriteLine("");  
    }  
    Console.WriteLine("=====");  
}
```

不規則陣列的子陣列

```
[0][0,0] = 5  
[0][0,1] = 6
```

```
[0][1,0] = 7  
[0][1,1] = 8
```

=====

```
[1][0,0] = 10  
[1][0,1] = 20  
[1][0,2] = 30
```

```
[1][1,0] = 40  
[1][1,1] = 50  
[1][1,2] = 60
```

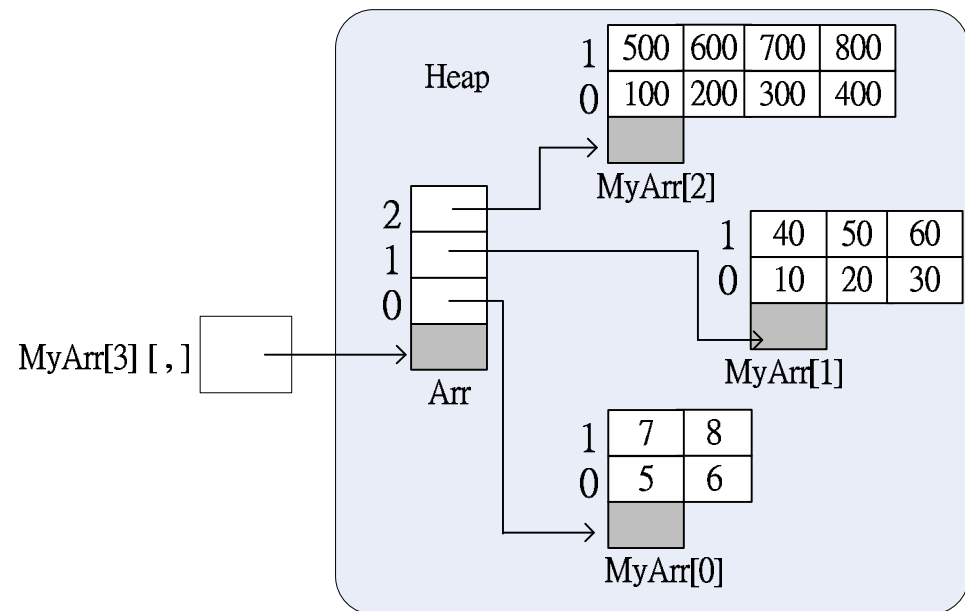
=====

```
[2][0,0] = 100  
[2][0,1] = 200  
[2][0,2] = 300  
[2][0,3] = 400
```

```
[2][1,0] = 500  
[2][1,1] = 600  
[2][1,2] = 700  
[2][1,3] = 800
```

=====

請按任意鍵繼續 . . .



foreach 陳述式用於不規則陣列

不規則陣列

```
int sum = 0;
int[][] myarr = new int[3][];
myarr[0] = new int[] { 10, 20 };
myarr[1] = new int[] { 30, 40, 50 };
myarr[2] = new int[] { 60, 70, 80, 90 };
```

```
foreach (int[] array in myarr)
{
    Console.WriteLine("新陣列開始");
    foreach (int item in array)
    {
        sum += item;
        Console.WriteLine("項目:{0}, 目前總和: {1}", item, sum);
    }
}
```

```
新陣列開始
項目:10, 目前總和: 10
項目:20, 目前總和: 30
新陣列開始
項目:30, 目前總和: 60
項目:40, 目前總和: 100
項目:50, 目前總和: 150
新陣列開始
項目:60, 目前總和: 210
項目:70, 目前總和: 280
項目:80, 目前總和: 360
項目:90, 目前總和: 450
請按任意鍵繼續 . . .
```

不規則陣列和矩形陣列的比較

以下範例顯示不規則陣列 (jagged array) 和矩形陣列 (rectangular array) 的差異：

- 都有九個整數元素，但儲存結構不同
- 矩形陣列有一個單一的陣列物件，而不規則陣列有四個陣列物件。

