第八章
陣列與字串

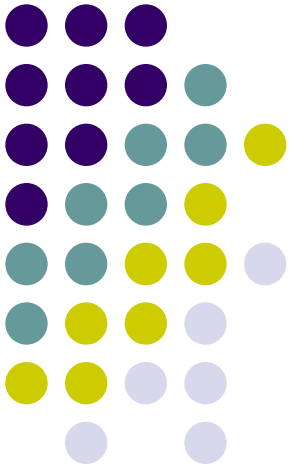
認識一維與二維以上的陣列

瞭解陣列元素的表示方法

迴圈與陣列的使用

學習傳遞陣列至函數裡

字串的認識與使用



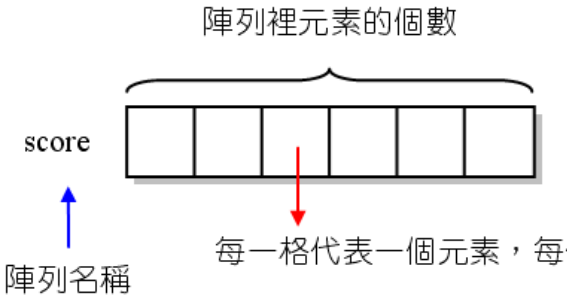
一維陣列的宣告 (1/2)

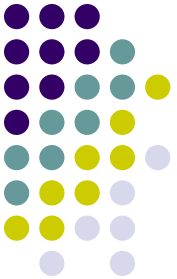
- 陣列可用來存放相同型態的元素
- 一維陣列的宣告格式如下所示

```
資料型態 陣列名稱[個數];     // 宣告一維陣列
```

- 下面的範例都是合法的一維陣列宣告

```
int score[6];
float temp[7];
char name[12];
```





一維陣列的宣告 (2/2)

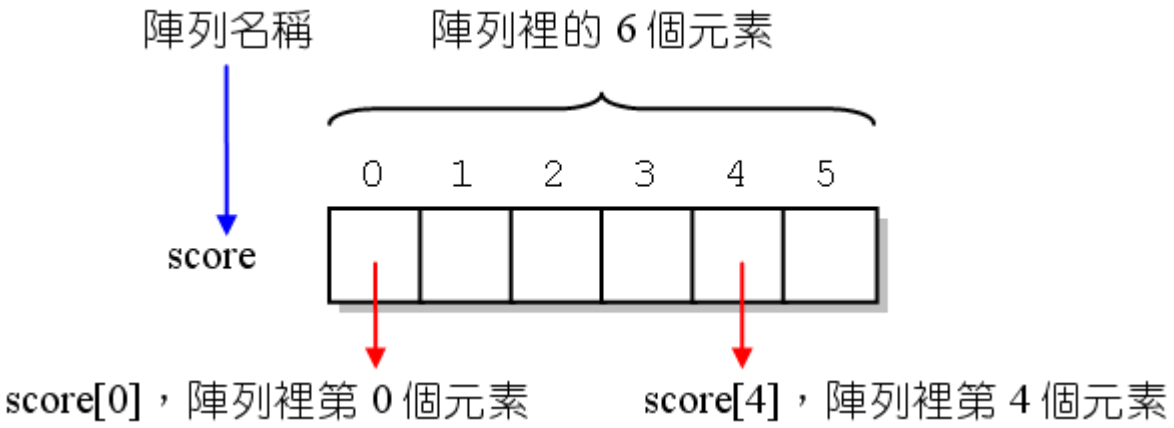
- 利用sizeof() 印出陣列score與其中任一個元素的長度

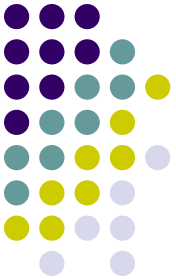
```
01  // prog8_1, 一維陣列
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int score[6];
08
09     // 印出陣列中個別元素的長度及陣列的總長度
10     cout << "sizeof(score[1])=" << sizeof(score[1]) << endl;
11     cout << "sizeof(score)=" << sizeof(score) << endl;
12     system("pause");
13     return 0;
14  }
```

```
/* prog8_1 OUTPUT------

sizeof(score[1])=4
sizeof(score)=24

----------------------*/
```

陣列元素的表示方法

- 下圖為score陣列中元素的表示法及排列方式





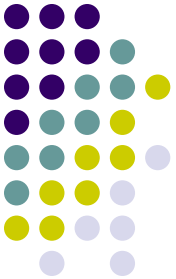
陣列初值的設定 (1/2)

- 在宣告時就給予陣列初值，可用下面的語法

資料型態 陣列名稱[n]={初值 0,初值 1,...,初值 n-1};

- C陣列宣告及初值的設定範例

```
int day[12]={31,28,31,30,31,30,31,31,30,31,30,31};

int day[]={31,28,31,30,31,30,31,31,30,31,30,31};

int data[5]={100}; // 將陣列 data 內的所有元素值都設為 100
```

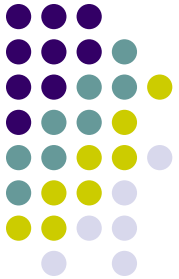


陣列初值的設定 (2/2)

- **prog8_2是一維陣列設定初值的範例**

```
01  // prog8_2, 一維陣列的設值
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int i,a[]={15,6,8};
08     int length=sizeof(a)/sizeof(int);          // 計算陣列元素個數
09     for(i=0;i<length;i++)                      // 印出陣列的內容
10        cout << "a[" << i << "]=" << a[i] << ", ";
11     cout << endl << "array a has " << length << " elements";  // 印出 length
12     system("pause");
13     return 0;
14  }
```

```
/* prog8_2 OUTPUT------

a[0]=15, a[1]=6, a[2]=8,
array a has 3 elements

----------------------*/
```

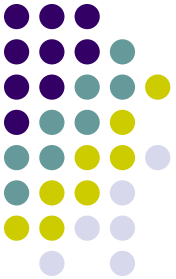


簡單的範例

- 下面的例子說明如何將陣列裡的最大及最小值列出

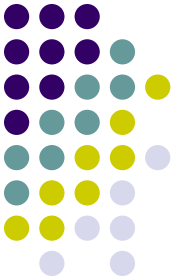
```
01  // prog8_3, 比較陣列元素值的大小
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int A[]={48,75,30,17,62};
08     int i,min=A[0],max=A[0];
09     int length=sizeof(A)/sizeof(int);    // 計算陣列元素個數
10     cout << "elements in array A are ";
11     for(i=0;i<length;i++)                // 印出陣列的內容
12     {
13        cout << A[i] << "  ";
14        if(A[i]>max)                      // 判斷最大值
15           max=A[i];
16        if(A[i]<min)                      // 判斷最小值
17           min=A[i];
18     }
19     cout << endl << "Maximum is " << max;            // 印出最大值
20     cout << endl << "Minimum is " << min << endl;    // 印出最小值
21     system("pause");
22     return 0;
23  }
```

```
/* prog8_3 OUTPUT-----------------------

elements in array A are 48  75  30  17  62
Maximum is 75
Minimum is 17

---------------------------------------*/
```



陣列界限的檢查 (1/3)

- C++並不會檢查註標值的大小
- 當註標值超過陣列的長度時，可能造成不可預期的錯誤
- 這種錯誤是在執行時才發生的（run-time error），而不是在編譯時期發生的錯誤（compile-time error），編譯程式無法提出任何的警告訊息

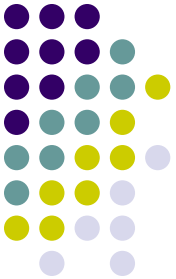


陣列界限的檢查 (2/3)

- 下面的程式裡，將陣列界限的檢查範圍加入

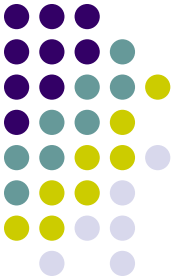
```
01  // prog8_4, 陣列的界限檢查
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  #define MAX 5
06  int main(void)
07  {
08     int score[MAX];
09     int i=0,num;
10     float sum=0.0f;
11     cout << "Enter 0 stopping input!!" << endl;
```

```
/* prog8_4 OUTPUT--------
Enter 0 stopping input!!
Input score:68
Input score:93
Input score:84
Input score:71
Input score:63
No more space!!
Average of all is 75.8
------------------------*/
```



陣列界限的檢查 (3/3)

```
12      do
13      {
14         if(i==MAX)              // 當 i 的值為 MAX，表示陣列已滿，即停止輸入
15         {
16            cout << "No more space!!" << endl;
17            i++;
18            break;
19         }
20         cout << "Input score:";
21         cin >> score[i];
22      }while(score[i++]>0);       // 輸入成績，輸入 0 或負數時結束
23      num=i-1;
24      for(i=0;i<num;i++)
25         sum+=score[i];           // 計算平均成績
26      cout << "Average of all is " << sum/num << endl;
27      system("pause");
28      return 0;
29   }
```



二維陣列的宣告 (1/4)

- 二維陣列（2-dimensional array）宣告格式

資料型態 陣列名稱 [列的個數] [行的個數];

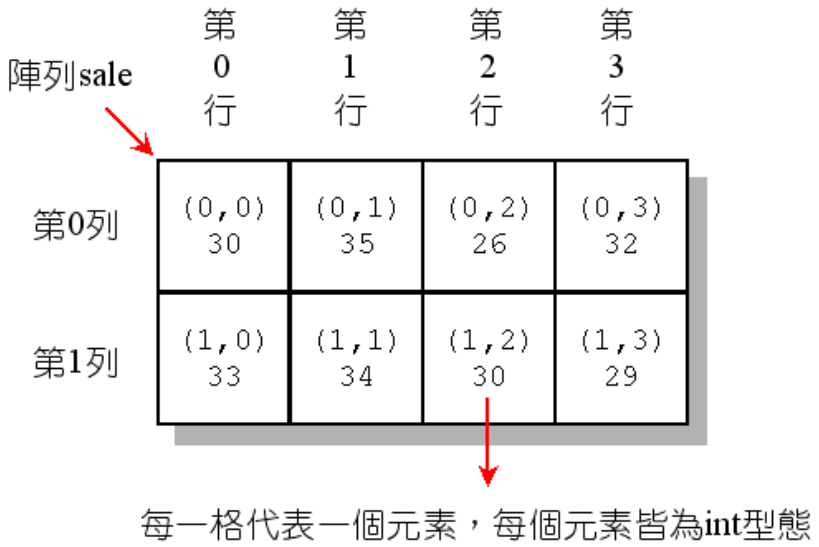
- 下面的範例都是合法的陣列宣告

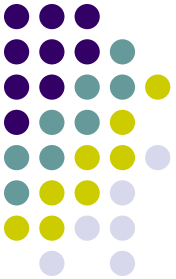
```
int data[6][5];       // 宣告整數陣列 data，元素個數為 6*5=30
float score[3][7];    // 宣告浮點數陣列 score，元素個數為 3*7=21
```

二維陣列的宣告 (2/4)

- 下表是年度銷售量，可利用二維陣列將資料儲存起來

業務員	本年度銷售量			
	第一季	第二季	第三季	第四季
1	30	35	26	32
2	33	34	30	29





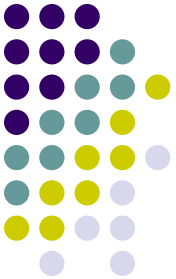
二維陣列的宣告 (3/4)

- 想直接在宣告時就給予陣列初值，如下面的格式

```
資料型態 陣列名稱[列的個數][行的個數]={{ 第 0 列初值 },
                                        { 第 1 列初值 },
                                        {    ...     },
                                        { 第 n 列初值 }};
```

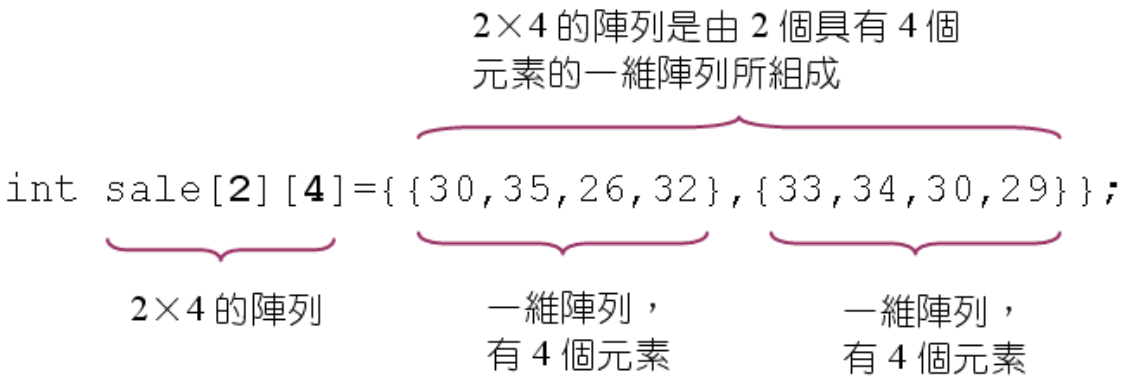
- 下面的陣列sale宣告及初值的設定範例

```
int sale[2][4]={{30,35,26,32},     // 二維陣列的初值設定
                {33,34,30,29}};
```



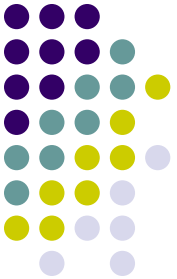
二維陣列的宣告 (4/4)

- 二維陣列的初值設定可依下面的說明來設定



- 宣告二維陣列，並設定初值時，列的個數可以省略

```
int temp[][4]={{30,35,26,32},
               {33,34,30,29},
               {25,33,29,25}};
```



二維陣列元素的引用及存取

- 以二維陣列sale為例，介紹如何讀取二維陣列

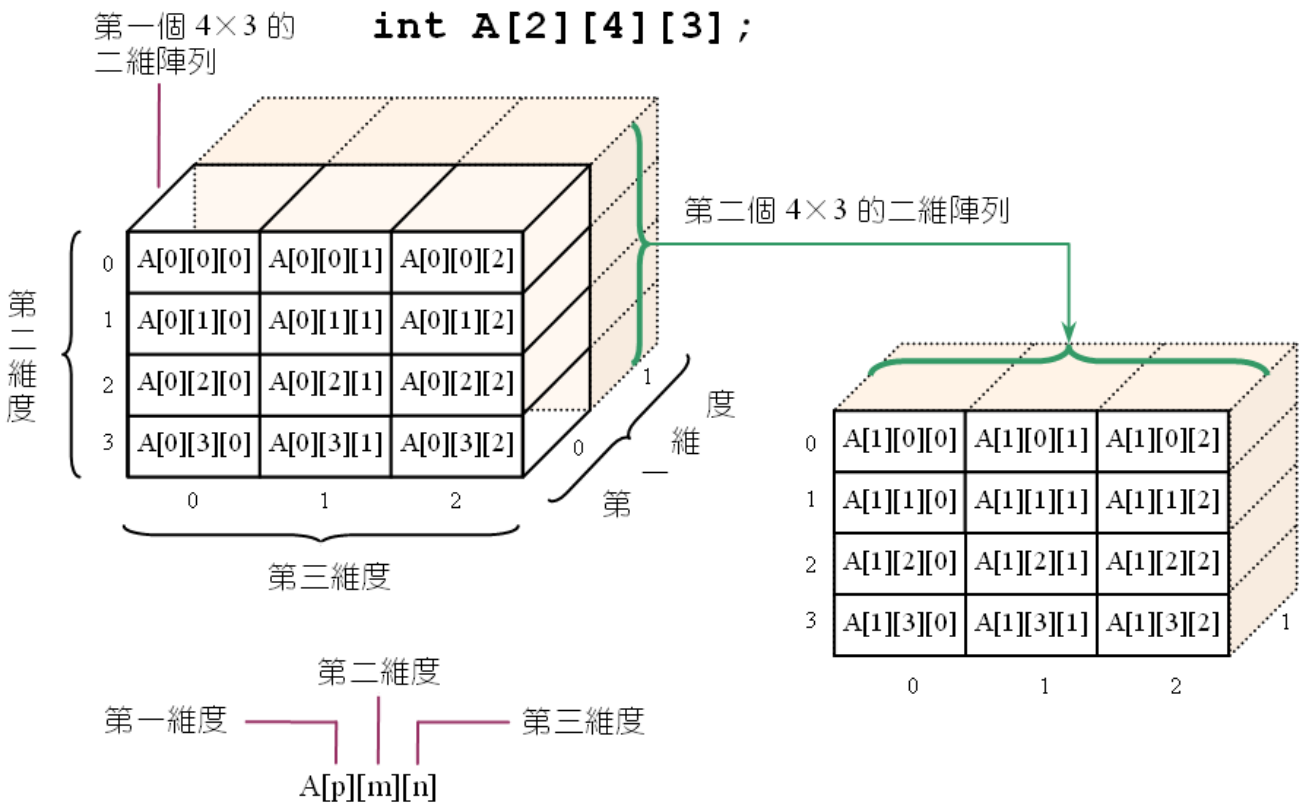
```
01  // prog8_5, 二維陣列的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int i,j,sum=0;
08     int sale[2][4]={{30,35,26,32},{33,34,30,29}};// 宣告陣列並設定初值
09     for(i=0;i<2;i++)                  // 輸出銷售量並計算總銷售量
10     {
11        cout << "業務員" << (i+1) << "的業績分別為 ";
12        for(j=0;j<4;j++)
13        {
14           cout << sale[i][j] << " ";
15           sum+=sale[i][j];
16        }
17        cout << endl;
18     }
19     cout << endl << "本年度總銷售量為" << sum << "輛車" << endl;
20     system("pause");
21     return 0;
22  }
```

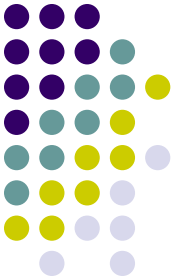
```
/* prog8_5 OUTPUT-----------

業務員 1 的業績分別為 30 35 26 32
業務員 2 的業績分別為 33 34 30 29

本年度總銷售量為 249 輛車

---------------------------*/
```

多維陣列 (1/4)

- 要宣告一個2×4×3的陣列A，可以利用下面的語法

```
int A[2][4][3];                    // 宣告 2×4×3 整數陣列 A
```





多維陣列 (2/4)

- 下面的範例說明如何在三維陣列裡，找出元素的最大值

```
01  // prog8_6, 三維陣列的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int A[2][4][3]={{{21,32,65},          // 宣告陣列並設定初值
08                      {78,94,76},
09                      {79,44,65},
10                      {89,54,73}},
11                     {{32,56,89},
12                      {43,23,32},
13                      {32,56,78},
14                      {94,78,45}}};
15     int i,j,k,max=A[0][0][0];             // 設定 max 為 A 陣列的第一個元素
16
```

設定 2×4×3 陣列的初值

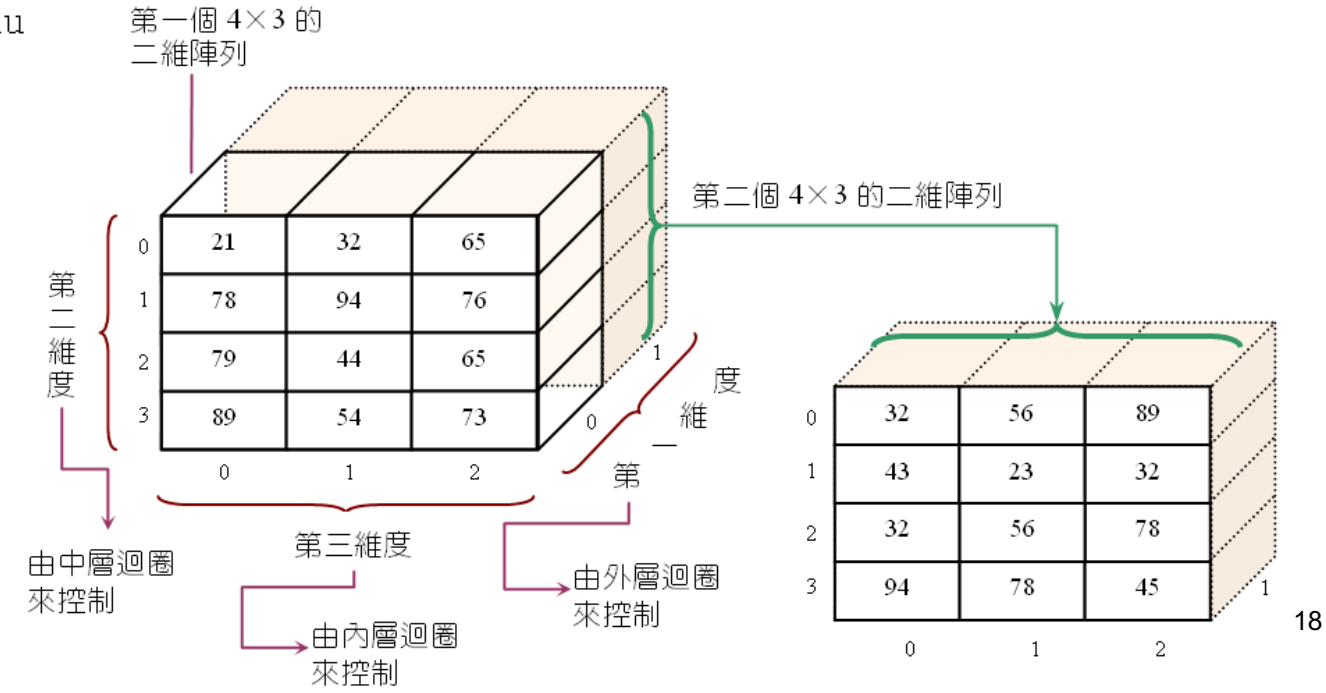
```
/* prog8_6 OUTPUT---
max=94
--------------------*/
```

多維陣列 (3/4)

```
/* prog8_6 OUTPUT---
max=94
--------------------*/
```

```
17      for(i=0;i<2;i++)
18         for(j=0;j<4;j++)
19            for(k=0;k<3;k++)
20               if(max<A[i][j][k])
21                  max=A[i][j][k];
22      cout << "max=" << max << endl;      // 印出陣列的最大值
23
24      system("pau
25      return 0;
26   }
```

利用三個 for 迴圈找出陣列的最大值



多維陣列 (4/4)

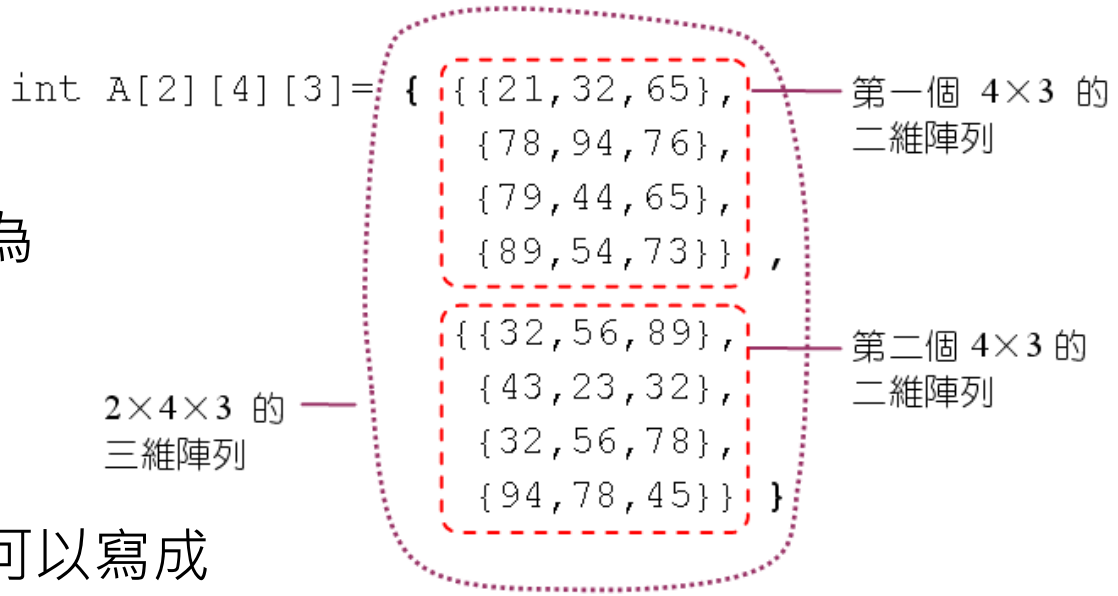
- 三維陣列A的第一個4×3的二維陣列為

```
{{21,32,65},
 {78,94,76},
 {79,44,65},
 {89,54,73}}
```

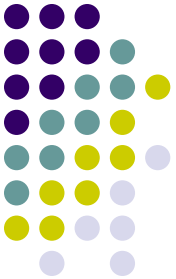
第二個4×3的二維陣列為

```
{{32,56,89},
 {43,23,32},
 {32,56,78},
 {94,78,45}}
```

因此2×4×3的三維陣列可以寫成



2×4×3 的三維陣列 = { 4×3 的二維陣列，4×3 的二維陣列 }

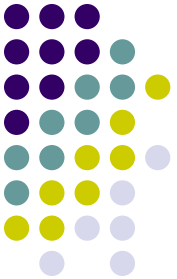


以一維陣列為引數來傳遞 (1/3)

- 下面為傳遞一維陣列至函數的格式

```
傳回值型態 函數名稱(資料型態 []);        // 宣告函數原型
int main(void)
{
    資料型態 陣列名稱[個數];
        ...
    函數名稱(陣列名稱);
        ...
}

傳回值型態 函數名稱(資料型態 陣列名稱[] )
{
        ...
}
```

中括號內可以不填入元素的個數

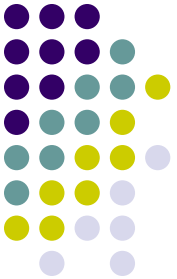


以一維陣列為引數來傳遞 (2/3)

- 下面的程式是以一維陣列為引數，傳遞到函數的範例

```
01  // prog8_7, 以一維陣列為引數
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  #define SIZE 5
06  void show(int []);                          // 函數原型的宣告
07  double average(int []);                     // 函數原型的宣告
08  int main(void)
09  {
10     int score[SIZE]={89,54,73,95,71};        // 宣告陣列並設定初值
11     cout << "學生的成績為 ";
12     show(score);
13     cout << "平均成績=" << average(score) << endl;
14
15     system("pause");
16     return 0;
17  }
18
```

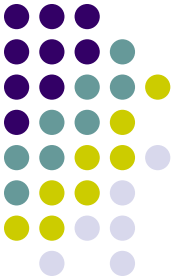
```
/* prog8_7 OUTPUT--------
學生的成績為 89 54 73 95 71
平均成績=76.4
------------------------*/
```



以一維陣列為引數來傳遞 (3/3)

```
19  void show(int a[])                          // 顯示學生成績
20  {
21     for(int i=0;i<SIZE;i++)
22        cout << a[i] << " ";
23     cout << endl;
24     return;
25  }
26
27  double average(int a[])                     // 計算平均成績
28  {
29     double sum=0;
30     for(int i=0;i<SIZE;i++)
31        sum+=a[i];
32     return (sum/SIZE);
33  }
```

```
/* prog8_7 OUTPUT--------
學生的成績為 89 54 73 95 71
平均成績=76.4

------------------------*/
```



傳遞多維陣列 (1/2)

- 傳遞二維陣列至函數的格式

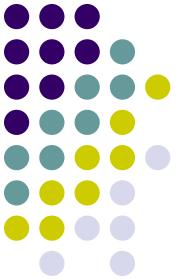
```
傳回值型態 函數名稱(資料型態 [列的個數][行的個數]);      // 宣告函數原型
int main(void)
{
   資料型態 陣列名稱[列的個數][行的個數];
      ...
   函數名稱(陣列名稱);
      ...
}

傳回值型態 函數名稱(資料型態 陣列名稱[列的個數][行的個數])
{
      ...
}
```

中括號內可以不填入列的個數

中括號內可以不填入列的個數

中括號內必須填入行的個數

傳遞多維陣列 (2/2)



- 下面是傳遞二維陣列到函數的練習

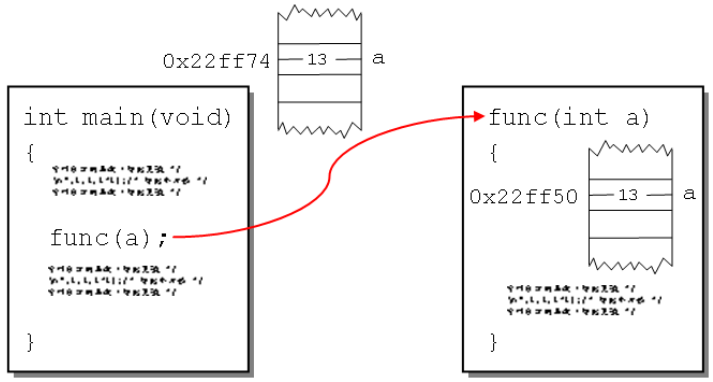
```
01  // prog8_8, 傳遞二維陣列
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  #define LEN 2
06  #define WID 5
07  void show(int [LEN][WID]);                    // 函數原型的宣告
08  int main(void)
09  {
10     int A[LEN][WID]={{81,52,13,96,27},         // 宣告陣列並設定初值
11                      {24,23,10,32,16}};
12     show(A);
13
14     system("pause");
15     return 0;
16  }
17
18  void show(int a[LEN][WID])                    // 顯示陣列內容
19  {
20     for(int i=0;i<LEN;i++)
21     {
22        for(int j=0;j<WID;j++)
23           cout << a[i][j] << " ";
24        cout << endl;
25     }
26     return;
27  }
```

```
/* prog8_8 OUTPUT---

81 52 13 96 27
24 23 10 32 16

--------------------*/
```

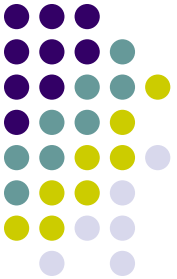
傳「值」還是傳「位址」？(1/3)

- 下面的程式說明函數傳值的過程

```
01  // prog8_9, 印出變數的位址
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  void func(int);              // 宣告函數原型
06  int main(void)
07  {
08     int a=13;
09     cout << "In main(),a=" << a << ",address=" << &a << endl;
10     func(a);
11     system("pause");
12     return 0;
13  }
14
15  void func(int a)             // 自訂函數 func()
16  {
17     cout << "In func(),a=" << a << ",address=" << &a << endl;
18     return;
19  }
```




```
/* prog8_9 OUTPUT---------------
In main(),a=13,address=0x22ff74
In func(),a=13,address=0x22ff50
--------------------------------*/
```



傳「值」還是傳「位址」？(2/3)

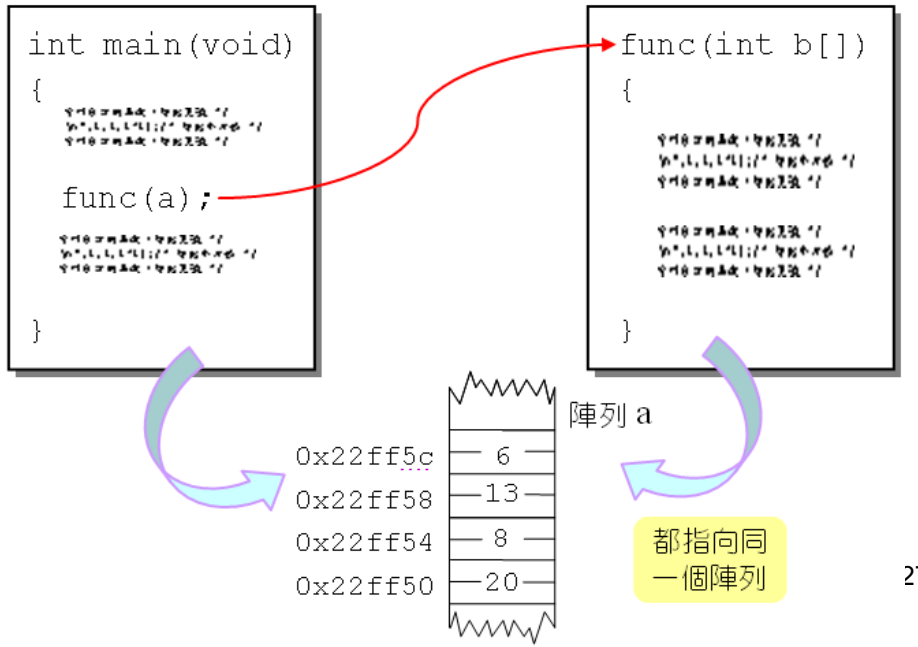
- 下面是函數傳遞陣列位址的範例

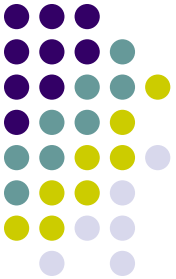
```
01  // prog8_10, 印出陣列的位址
02  #include <iostream>
03  #include <cstdlib>
04  #include <iomanip>
05  using namespace std;
06  void func(int []);                              // 宣告函數原型
07  int main(void)
08  {
09     int i,a[4]={20,8,13,6};
10     cout << "In main()," << endl;                // 印出陣列 a 的值及位址
11     for(i=0;i<4;i++)
12     {
13        cout << "a[" << i << "]=" << setw(2) << a[i];
14        cout << ",address=" << &a[i] << endl;
15     }
16     func(a);
17     system("pause");
18     return 0;
19  }
20
```

傳「值」還是傳「位址」？(3/3)

```
21  void func(int b[])                          // 自訂函數 func()
22  {
23     int i;
24     cout << "In func()," << endl;            // 印出陣列 b 的值及位址
25     for(i=0;i<4;i++)
26     {
27        cout << "b[" << i << "]=" << setw(2) << b[i];
28        cout << ",address=" << &b[i] << endl;
29     }
30     return;
31  }
```

```
/* prog8_10 OUTPUT-------
In main(),
a[0]=20,address=0x22ff50
a[1]= 8,address=0x22ff54
a[2]=13,address=0x22ff58
a[3]= 6,address=0x22ff5c
In func(),
b[0]=20,address=0x22ff50
b[1]= 8,address=0x22ff54
b[2]=13,address=0x22ff58
b[3]= 6,address=0x22ff5c

------------------------*/
```



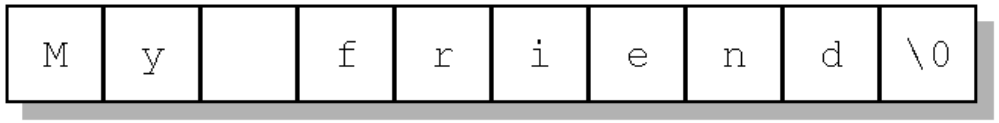


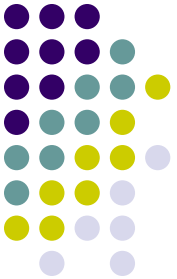
字串常數

- 字串常數是以兩個雙引號（"）包圍起來的資料

```
"Dev C++"
"Merry Christmas!"
"Computer"
```

- 字串儲存在記憶體時，會加上字串結束字元\0做結尾





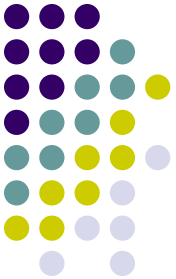
字串的宣告與初值的設定 (1/2)

- 字串的宣告格式如下

```
char 字元陣列名稱[字串長度];
char 字元陣列名稱[字串長度]="字串常數";
```

- 下面的範例為合法的字串變數宣告

```
char mystr[30];                  // 宣告字元陣列 mystr，長度為 30 個字元
char name[15]="Tippi Hong";      // 宣告字元陣列 name，初值為 Tippi Hong
```

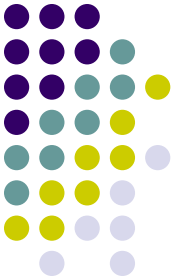


字串的宣告與初值的設定 (2/2)

- 下面的程式可印出字元及字串的長度

```
01  // prog8_11, 印出字元及字串的長度
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     char a[]="My friend";
08     char b='c',str[]="c";
09     cout << "sizeof(a)=" << sizeof(a) << endl;
10     cout << "sizeof(b)=" << sizeof(b) << endl;
11     cout << "sizeof(str)=" << sizeof(str) << endl;
12     system("pause");
13     return 0;
14  }
```

```
/* prog8_11 OUTPUT---

sizeof(a)=10
sizeof(b)=1
sizeof(str)=2

--------------------*/
```



字串的輸出與輸入 (1/3)

- 以cout輸出字串常數，須用資料流插入運算子「<<」

```
cout << "It is a windy day!" << endl;
```

- 利用cout印出字串物件的內容

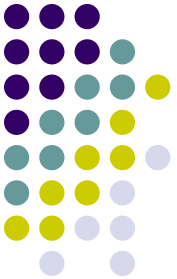
```
char str[20]="Time is money";      // 宣告字串 str 並設值
cout << "str=" << str;             // 印出 str 的內容
```

- 以cin輸入字串時，要使用資料流擷取運算子「>>」

```
char str[20]      // 宣告字串 str
cin >> str;       // 由鍵盤中讀取字串給 str 存放
```

- 使用cin輸入資料內容前，會利用cout輸出提示訊息

```
cout << "Input a string:";     // 提示訊息，請使用者輸入資料
cin >> str;                    // 由鍵盤中讀取字串給 str 存放
```

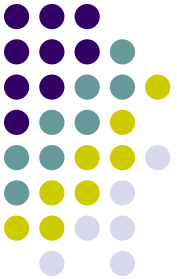


字串的輸出與輸入 (2/3)

- 使用cout及cin的範例（輸出有誤）

```
01  // prog8_12, 輸入及輸出字串
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     char name[15];
08     int i;
09     for(i=0;i<2;i++)
10     {
11        cout << "What's your name? ";
12        cin >> name;                    // 以 cin 輸入字串
13        cout << "Hi, " << name << ", how are you?" << endl << endl;
14     }
15     system("pause");
16     return 0;
17  }
```

```
/* prog8_12 OUTPUT----------

What's your name? Tippi
Hi, Tippi, how are you?

What's your name? Alice Wu
Hi, Alice, how are you?

--------------------------*/
```

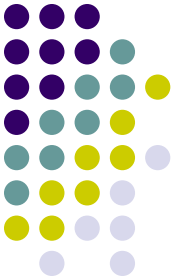


字串的輸出與輸入 (3/3)

- 利用cin.getline() 修正prog8_12可能出現的錯誤

```
01  // prog8_13, 修正 prog8_12 可能出現的錯誤
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     char name[15];
08     int i;
09     for(i=0;i<2;i++)
10     {
11        cout << "What's your name? ";
12        cin.getline(name,15);          // 以 cin.getline()輸入字串
13        cout << "Hi, " << name << ", how are you?" << endl << endl;
14     }
15     system("pause");
16     return 0;
17  }
```

```
/* prog8_13 OUTPUT------------

What's your name? Lucy Wang
Hi, Lucy Wang, how are you?

What's your name? Minnie Hong
Hi, Minnie Hong, how are you?

-----------------------------*/
```



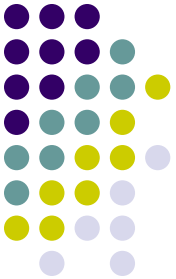
cin.get()

- 輸入單一字元的情況下，可使用cin.get()，格式如下

```
cin.get(字元變數名稱);
```

- 舉例來說

```
char ch;          // 宣告字元變數 ch
cin.get(ch);      // 由鍵盤輸入一個字元，並指定給 ch 存放
```

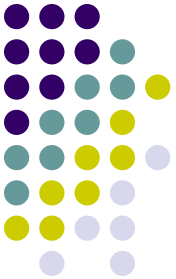


混合輸入的問題 (1/2)

- 字串與數值混合輸入時可能會發生問題，如下面的程式

```
01  // prog8_14, 字串與數值混合輸入
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int age;
08     char name[20];
09     cout << "How old are you? ";
10     cin >> age;
11     cout << "What's your name? ";
12     cin.getline(name,20);
13     cout << name << " is " << age << "-years-old!" << endl;
14     system("pause");
15     return 0;
16  }
```

```
/* prog8_14 OUTPUT-----------------

How old are you? 18
What's your name?  is 18-years-old!

---------------------------------*/
```



混合輸入的問題 (2/2)

- 於prog8_14中，多加一行cin.get(); 即可修正錯誤：

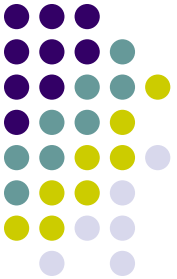
```
10    cin >> age;
11    cin.get();                          // 接收多餘的\n
12    cout << "What's your name? ";
```

或是將上面2行敘述寫成一行：

```
(cin >> age).get();
```

經過更改後的程式執行結果如下所示：

```
/* prog8_14 OUTPUT------------
How old are you? 18
What's your name? Tippi Hong
Tippi Hong is 18-years-old!
-----------------------------*/
```



C++型態字串

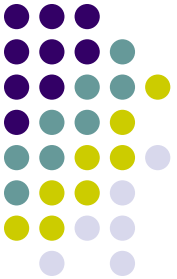
- 使用基本資料型態宣告的，稱為變數（variable）
- 在物件導向程式設計（object oriented programming）裡以類別宣告的，稱為「物件」（object）
- string類別宣告的就是字串，宣告格式

```
string 字串名稱="字串常數";
```

```
string 字串名稱;
字串名稱="字串常數";
```

- 下面的範例為合法的字串宣告

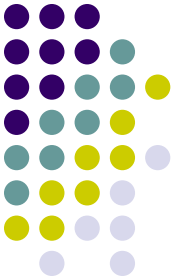
```
string str1;                    // 宣告 string 類別物件 str1
str1="Hello C++!";              // 為 str1 設值為"Hello C++!"

string str2="Hello C++!";       // 宣告 string 類別物件 str2，並直接設值

string str3="";                 // 宣告 string 類別物件 str3，並設值為空字串
```



C++型態的字串宣告

- 下表整理出常用的格式，並將該格式及對應的範例列出

格式	意義	範例解說
string 字串名稱("字串常數");	宣告 string 類別物件，並直接設值為括號裡的字串常數	string str("Time flies."); // str 的值為 Time flies.
string 字串名稱1(字串名稱2);	宣告名為字串名稱 1 的string類別物件，將其值設為括號裡的字串名稱 2 之值	string str1(str2); // str1 的值就等於 str2
string 字串名稱(n, '字元常數');	宣告名為字串名稱的string類別物件，將其初值設為 n 個字元常數	string str(6,'s'); // str 的值即為 ssssss

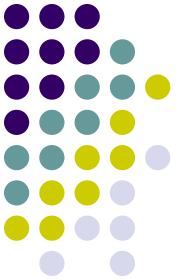


取得字串的長度 (1/2)

- length() 函數是string類別裡用來取得物件長度的函數，其用法如下

```
字串名稱.length();
```

句點是成員存取運算子（member access operator）

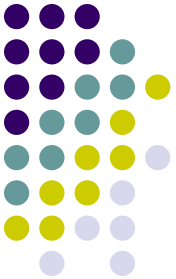


取得字串的長度 (2/2)

- 印出空字元陣列及空字串的長度

```
01  // prog8_15, 印出空字元陣列及空字串的長度
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  int main(void)
07  {
08     char str1[]="";
09     string str2;
10
11     cout << "str1=" << str1 << endl;
12     cout << "sizeof(str1)=" << sizeof(str1) << endl;
13     cout << "str2=" << str2 << endl;
14     cout << "length=" << str2.length() << endl;
15     system("pause");
16     return 0;
17  }
```

```
/* prog8_15 OUTPUT---
str1=
sizeof(str1)=1
str2=
length=0
----------------------*/
```



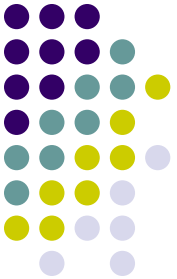
字串的輸出與輸入 (1/2)

- getline()的使用格式

```
getline（cin, 字串物件）;
```

- 想由使用者輸入含有空白的字串，可以寫出如下的敘述

```
getline(cin,str);     // 由鍵盤輸入字串，並指定給 str 存放
```

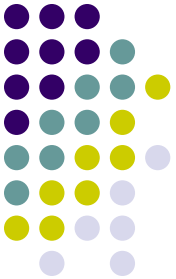


字串的輸出與輸入 (2/2)

- C++型態字串與數值混合輸入的範例如下：

```
01  // prog8_16, C++型態字串與數值混合輸入
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  int main(void)
07  {
08     int num;
09     string proverb;
10     cout << "輸入欲重複的次數: ";
11     (cin >> num).get();
12     cout << "輸入欲列印的字串: ";
13     getline(cin,proverb);
14     for(int i=1;i<=num;i++)
15        cout << proverb << endl;
16
17     system("pause");
18     return 0;
19  }
```

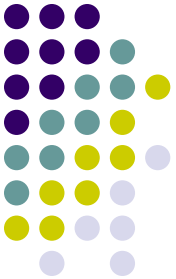
```
/* prog8_16 OUTPUT--------------------
輸入欲重複的次數: 3
輸入欲列印的字串: Practice makes perfect
Practice makes perfect
Practice makes perfect
Practice makes perfect
-------------------------------------*/
```



字串的運算 (1/2)

- 常用的字串運算子

運算子	範例	說 明
+	str1+str2	合併字串 str1 與 str2
=	str1=str2	將 str2 的值指定給 str1 存放
+=	str1+=str2	合併字串 str1 與 str2，結果存放在 str1
>	str1>str2	兩個字串逐字元相比，相同時再比較下一個字元，直到字元不同時，即比較該字元的 ASCII 值，由此判斷 str1 是否大於 str2
>=	str1>=str2	以字元的 ASCII 值之順序，判斷 str1 是否大於等於 str2
<	str1<str2	以字元的 ASCII 值之順序，判斷 str1 是否小於 str2
<=	str1<=str2	以字元的 ASCII 值之順序，判斷 str1 是否小於等於 str2
==	str1==str2	以字元的 ASCII 值之順序，判斷 str1 是否等於 str2
!=	str1!=str2	以字元的 ASCII 值之順序，判斷 str1 是否不等於 str2

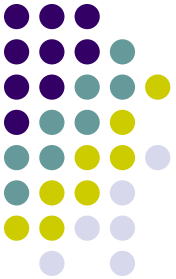


字串的運算 (2/2)

- 舉一個簡單的例子來說明字串的運算

```
01  // prog8_17, 字串的運算
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  int main(void)
07  {
08     string first="Junie";
09     string last="Hong";
10     cout << "full name=" << first+" "+last << endl;
11     first+=" ";                              // 字串 first 加上" "
12     first+=last;                             // 字串 first=first+last
13     cout << "full name=" << first << endl;
14
15     system("pause");
16     return 0;
17  }
```

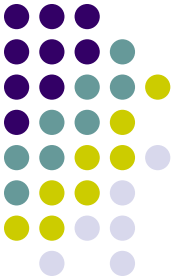
```
/* prog8_17 OUTPUT----

full name=Junie Hong
full name=Junie Hong

----------------------*/
```



字串類別裡的成員函數 (1/5)

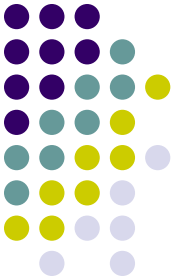
- 下面列出常用的字串處理函數

成 員 函 數	說 明
`str1.assign(str2)`	將 str2 的值指定給 str1 存放
`str1.assign(str2, index, length)`	從 str2 的第 index 個字元取出 length 個字元指定給 str1 存放
`str1.at(index)`	從 str1 取出第 index 個字元，若 index 超過字串長度，即會立即終止取出的動作
`str1.append(str2)`	將 str2 附加在 str1 之後
`str1.append(str2, index, length)`	從 str2 的第 index 個字元開始，取出 length 個字元，附加在 str1 之後
`str1.erase(index, length)`	從 str1 的第 index 個字元開始，取出 length 個字元刪除



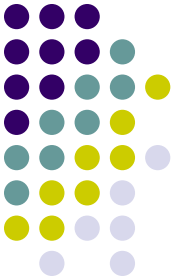
字串類別裡的成員函數 (2/5)

成 員 函 數	說 明
`str1.find(str2)`	於 str1 裡尋找 str2，並傳回 str2 在 str1 的位置
`str1.find(str2, index)`	從 str1 的第 index 個字元開始，尋找是否有 str2，並傳回 str2 在 str1 的位置
`str1.insert(index, str2)`	於 str1 的第 index 個字元開始，插入 str2
`str1.substr(index)`	取出從 str1 的第 index 開始，到字串結束為止的字元
`str1.substr(index, length)`	從 str1 的第 index 開始，取出 length 個字元
`str1.length()`	求取 str1 的長度
`str1.max_size()`	取出 str1 可使用的最大長度
`str1.empty()`	測試 str1 是否為空字串，若是，傳回 1（false），否則傳回 0（true）
`str1.clear()`	將 str1 的內容清除



字串類別裡的成員函數 (3/5)

成 員 函 數	說 明
`str1.swap(str2)`	將 str1 與 str2 的內容交換
`str1.compare(str2)`	將 str1 與 str2 相比，相同傳回 0，否則傳回 1
`str1.compare(str1_index, str1_length, str2, str2_index, str2_length)`	從 str1 的第 str1_index 個字元開始，取出長度為 str1_length 的子字串，與 str2 的第 str2_index 個字元開始，長度為 str2_length 的子字串之 ASCII 值相比。傳回值為 0，兩字串相等；小於 0，表示 str1 小於 str2；大於 0，str1 大於 str2
`str1.replace(index, length, str2)`	從 str1 的第 index 個字元開始，取出長度為 length 的子字串，以 str2 取代

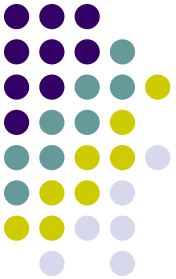


字串類別裡的成員函數 (4/5)

- 下面的範例是字串處理函數的運作

```
01  // prog8_18, 字串函數的練習
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  int main(void)
07  {
08     string str1="Hank ";
09     string str2="Wang";
10     string str3=", 2010/12/25";
11     cout << "str1=" << str1 << ", str2=" << str2;
12     cout << ", str3=" << str3 << endl;
```

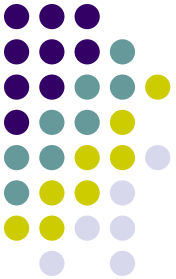
```
/* prog8_18 OUTPUT----------------------
str1=Hank , str2=Wang, str3=, 2010/12/25
執行 str1.append(str2)
str1=Hank Wang
執行 str1.append(str3,0,6)
str1=Hank Wang, 2010
取出 str1 第 5 個字元之後的子字串--> Wang, 2010
str1 長度=15
----------------------------------------*/
```



字串類別裡的成員函數 (5/5)

```
13      cout << "執行 str1.append(str2)" << endl;
14      str1.append(str2);
15      cout << "str1=" << str1 << endl;
16      cout << "執行 str1.append(str3,0,6)" << endl;
17      str1.append(str3,0,6);
18      cout << "str1=" << str1 << endl;
19      cout << "取出 str1 第 5 個字元之後的子字串--> ";
20      cout << str1.substr(5) << endl;
21      cout << "str1 長度=" << str1.length() << endl;
22
23      system("pause");
24      return 0;
25  }
```

```
/* prog8_18 OUTPUT----------------------
str1=Hank , str2=Wang, str3=, 2010/12/25
執行 str1.append(str2)
str1=Hank Wang
執行 str1.append(str3,0,6)
str1=Hank Wang, 2010
取出 str1 第 5 個字元之後的子字串--> Wang, 2010
str1 長度=15
-------------------------------------*/
```



C型態字串陣列 (1/4)

- 字串陣列的宣告及初值設定的格式如下

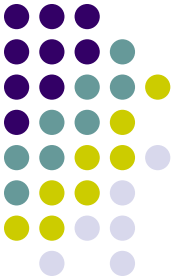
```
char 字串陣列名稱[陣列大小][字串長度];
```

或是在宣告陣列時直接設值

```
char 字串陣列名稱[陣列大小][字串長度]=
     {"字串常數 0", "字串常數 1",..., "字串常數 n"};
```

- 下面的範例為合法的字串陣列之宣告

```
char customer[6][15];
char students[3][10]={"David","Jane Wang","Tom Lee"};
```



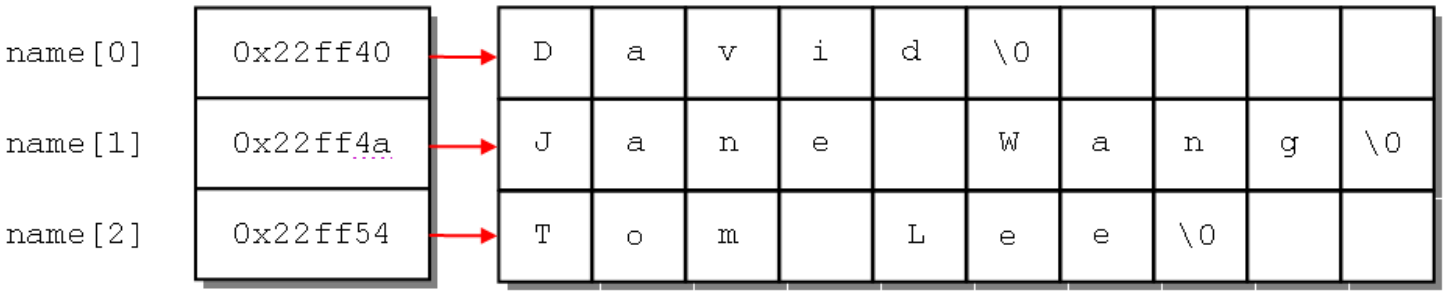
C型態字串陣列 (2/4)

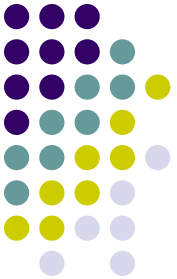
- 下面的程式印出字串陣列的內容

```
01  // prog8_19, 字串陣列
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int i;
08     char name[3][10]={"David","Jane Wang","Tom Lee"};
09     for(i=0;i<3;i++)                          // 印出字串陣列內容
10        cout << "name[" << i << "]=" << name[i] << endl;
11     cout << endl;
12     for(i=0;i<3;i++)                          // 印出字串陣列元素的位址
13     {
14        cout << "address of name[" << i << "]=" << &name[i] << endl;
15        cout << "address of name[" << i << "][0]=";
16        cout << (name+i) << endl << endl;
17     }
18
19     system("pause");
20     return 0;
21  }
```

C型態字串陣列 (3/4)

```
/* prog8_19 OUTPUT---------------

name[0]=David
name[1]=Jane Wang
name[2]=Tom Lee

address of name[0]=0x22ff40
address of name[0][0]=0x22ff40

address of name[1]=0x22ff4a
address of name[1][0]=0x22ff4a

address of name[2]=0x22ff54
address of name[2][0]=0x22ff54

--------------------------------*/
```



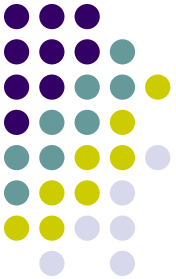


C型態字串陣列 (4/4)

- 下面的程式是練習字串陣列的輸入與輸出

```
01  // prog8_20, 字串陣列
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int i;
08     char students[3][15];
09     for(i=0;i<3;i++)
10     {
11        cout << "Input student" << i << "'s name:";
12        cin.getline(students[i],15);
13     }
14     cout << "***OUTPUT***" << endl;
15     for(i=0;i<3;i++)
16        cout << "students[" << i << "]=" << students[i] << endl;
17
18     system("pause");
19     return 0;
20  }
```

```
/* prog8_20 OUTPUT------------
Input student0's name:Mary Wang
Input student1's name:Queens
Input student2's name:Jerry Ho
***OUTPUT***
students[0]=Mary Wang
students[1]=Queens
students[2]=Jerry Ho
------------------------------*/
```



C++型態的字串陣列

- 下面的程式將字串陣列的內容複製到另一個字串陣列裡

```
01  // prog8_21, 字串陣列的複製
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  int main(void)
07  {
08     int i,j;
09     string students[3]={"David","Jane Wang","Tom Lee"};
10     string copystr[3];
11     for(i=0;i<3;i++)              // 將陣列 students 的內容複製到 copystr
12        copystr[i]=students[i];
13
14     for(i=0;i<3;i++)              // 印出陣列 copystr 的內容
15        cout << "copystr[" << i << "]=" << copystr[i] << endl;
16
17     system("pause");
18     return 0;
19  }
```

```
/* prog8_21 OUTPUT---

copystr[0]=David
copystr[1]=Jane Wang
copystr[2]=Tom Lee

--------------------*/
```


-The End-