C++程式設計

單元四：陣列與字串

陣列

陣列是一種重要的資料結構，可以幫助我們減少變數的命名。

當我們想使用大量相同性質的變數時，可以利用陣列的特性，以同一個變數名稱來代表這些資料，如此，可以讓減少大量的變數宣告。

而陣列資料的儲存時，會以連續的記憶體空間來存放。

一般變數和陣列變數表示方法比較

一般變數...

a1

a和1是一體的不可分割，所以使用時就是用a1。

陣列變數...

a[1]

陣列的名稱

陣列的索引，可以是一個指定的數字，也可以是一個變數，如：j=1，而a[j] 的意思就和a[1]的意思是一樣的。

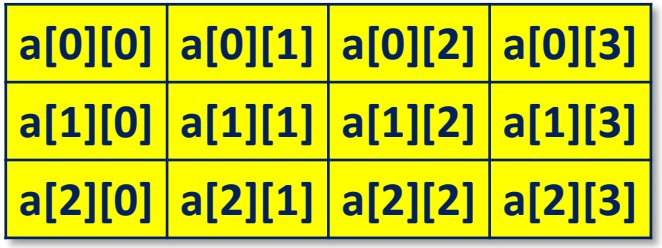
一維陣列在記憶體中儲存的情形

以 a 為名的字元(每筆資料占1Byte)陣列

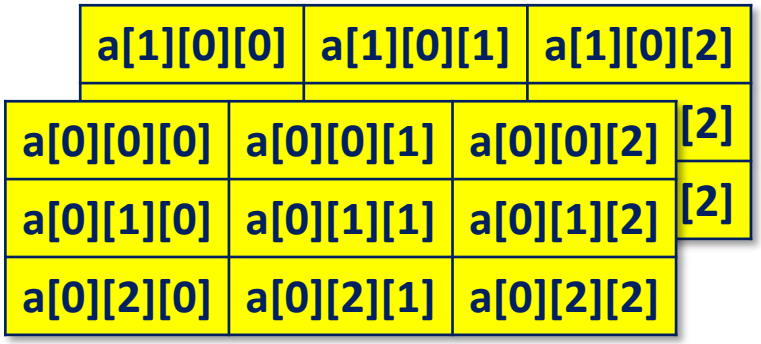
陣列的型態

依陣列的排列和佔用空間大小，可分為一維陣列、二維陣列.....等。

一維陣列



二維陣列



三維陣列

一維陣列宣告(一)

- 語法格式(一)：不指定初值

  資料型態 陣列名稱[資料筆數];

- 範例—

  int a[5];

- 說明—

  宣告名為 a 的陣列，該陣列內有五筆整數資料，分別為：

  a[0]、a[1]、a[2]、a[3]、a[4]

  因索引從0開始，因此，最大值為筆數-1

一維陣列宣告(二)

- 語法格式(二)：指定初值

```
資料型態 陣列名稱[]={n1,n2,n3…};
```

- 範例—

```
int a[]={3,4,1,2};
```

- 說明—

該陣列內有五筆整數資料，且分別指定每個元素的初值，該陣列之各元素為：

a[0]=3、a[1]=4、a[2]=1、a[3]=2

一維陣列資料的存取

因為陣列註標可以用變數來代替，因此，我們可以用迴圈的方式來操作陣列中的元素。

將陣列中的資料列出

```
int a[5];

for(i=0;i<5;i++){
    cout<<a[i];
}
```

將資料輸入到陣列中

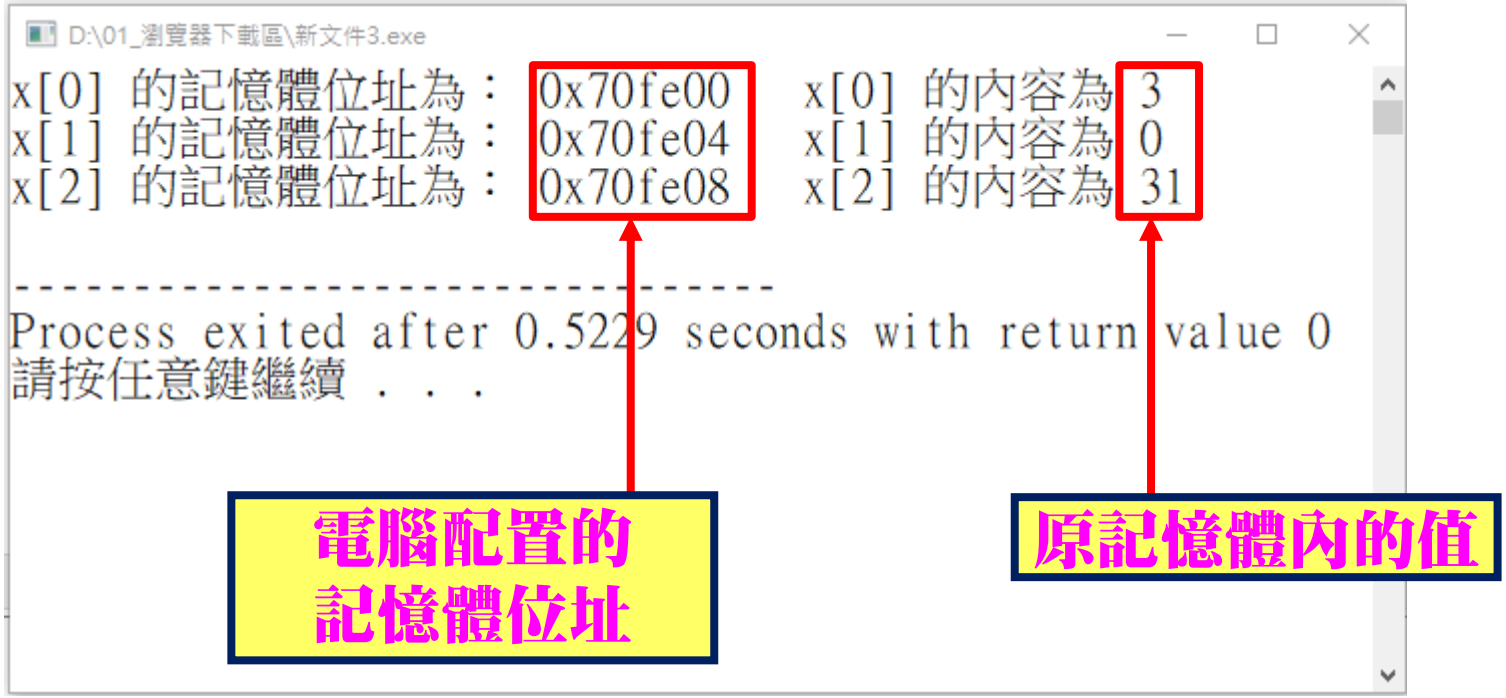
```
int a[5];

for(i=0;i<5;i++){
    cin>>a[i];
}
```

C++的陣列觀察(一)

```
#include <iostream>
#include <iomanip>
using namespace std;
main(){
	int a[3];
	int i;

	for(i=0;i<3;i++) {
		cout<<"a["<<i<<"] 的記憶體位址為： ";
		cout<<setw(8)<<&a[i]<<"\t";
		cout<<" a["<<i<<"] 的內容為 "<<a[i]<<endl;

	}
	return 0;
}
```

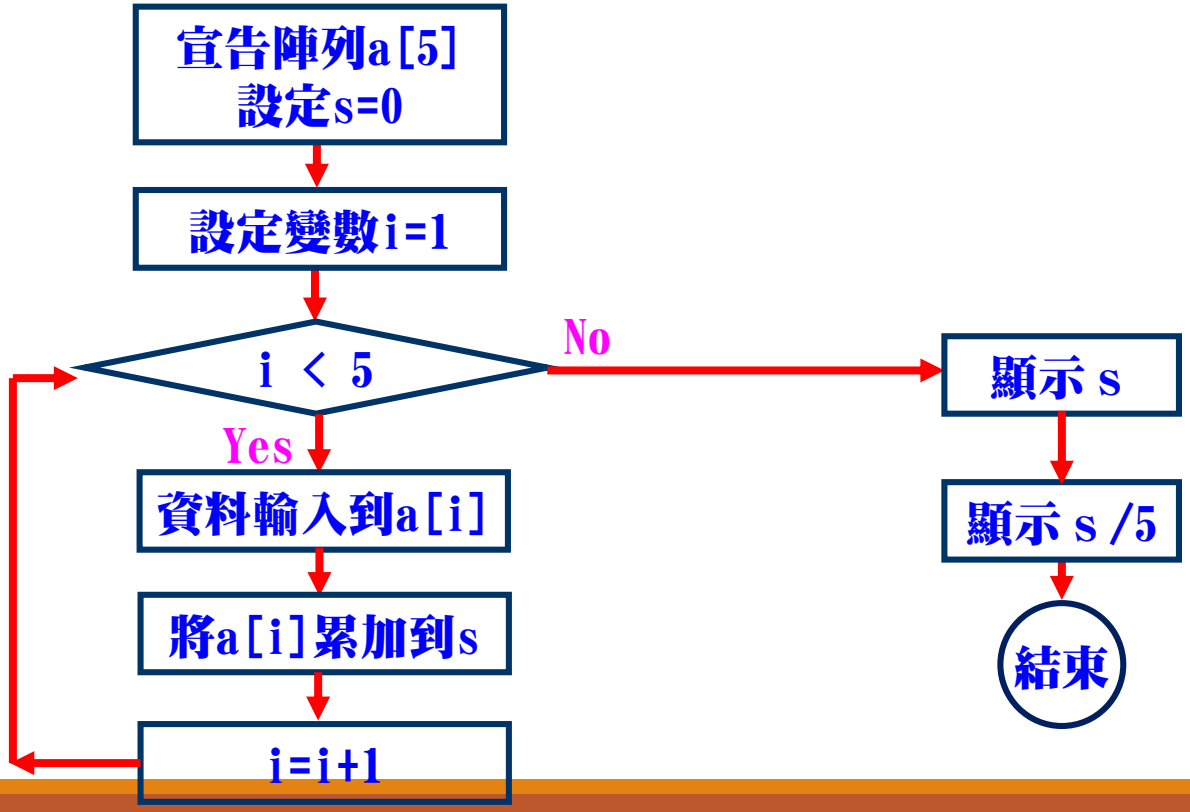
C++的陣列觀察(二)

執行結果：



陣列範例一(一)

以陣列儲存所輸入的五筆資料，並計算這五筆資料的和及平均值。



陣列範例一(二)

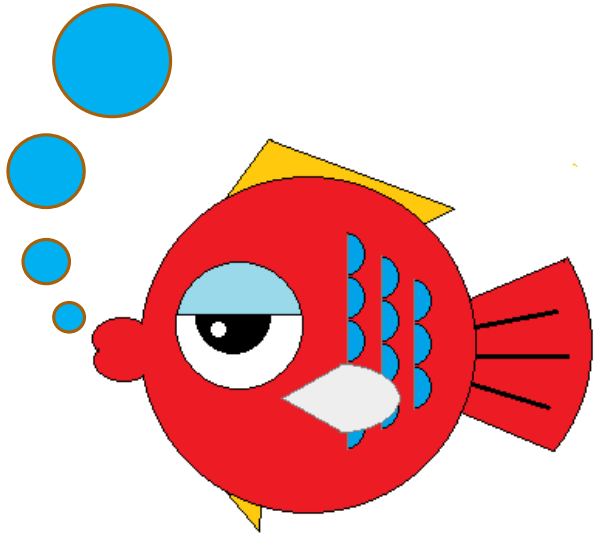
```
#include <iostream>
using namespace std;
main(){
	int a[5];
	int i,s;
	s=0;
	for(i=0;i<5;i++) {
		cout<<"請輸入第 "<< i +1<<" 筆資料： "; cin>>a[i];
		s=s+a[i];
	}
	cout<<"五個數的和為："<< s << endl;
	cout<<"五個數的平均為："<<  s/5.0  << endl;

	return 0;
 }
```

陣列最常見的應用—排序

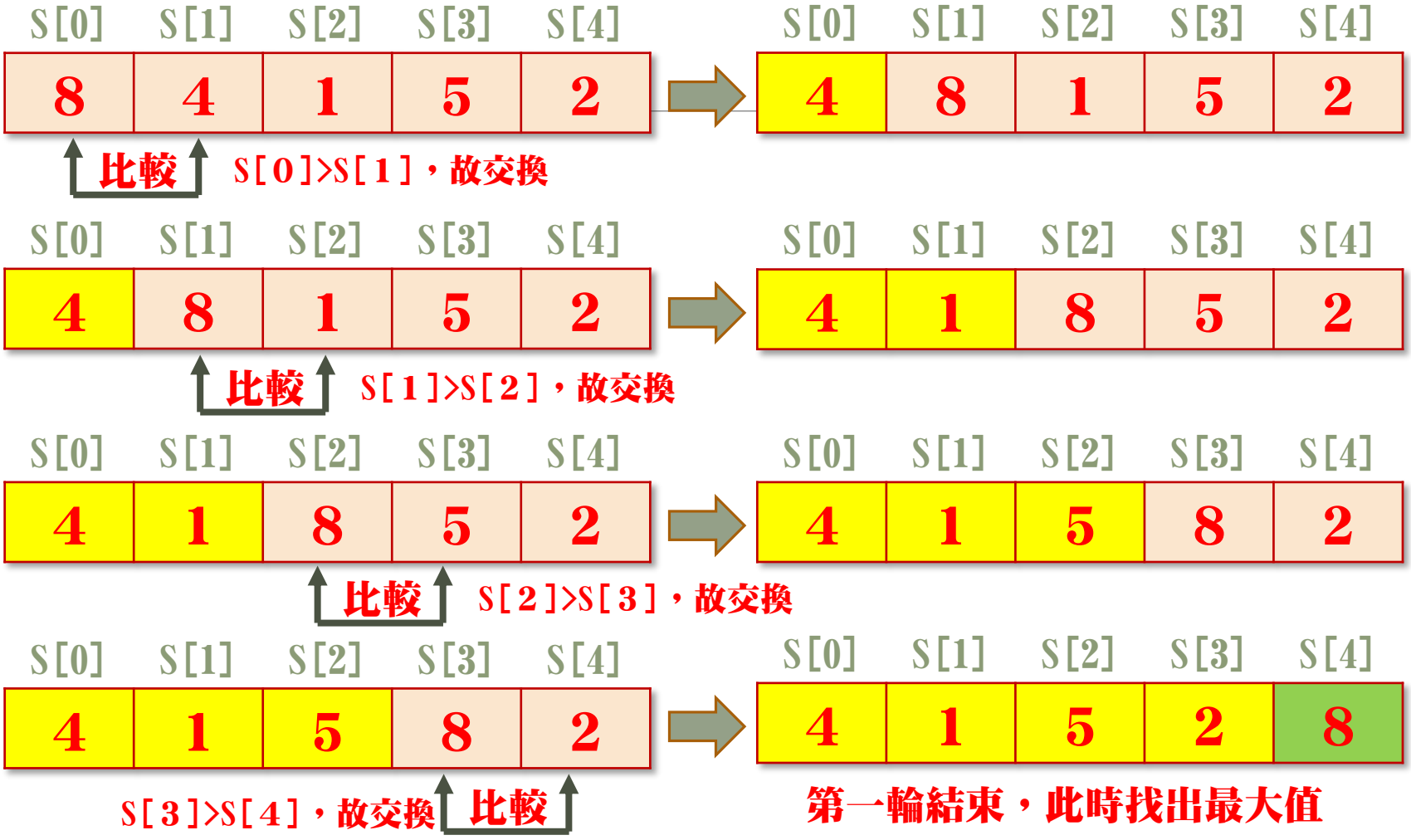
- 排序，是將一大小任意排列的數值依需求，將其由大到小或由小到大重新排列順序。
- 為方便排序，一般會將原數值放入陣列中，再以陣列存取的特性做排序。
- 排序的方式有很多種，常見的有泡沫（氣泡、交換）排序、選擇排序、二分排序……等。

泡沫排序

故名思義，猶如水中氣泡不斷的往水面上浮；排序的過程是將相鄰兩數不斷的做比較，依需求（大到小或小到大）來做兩數交換，直到所有的數比較交換為止。

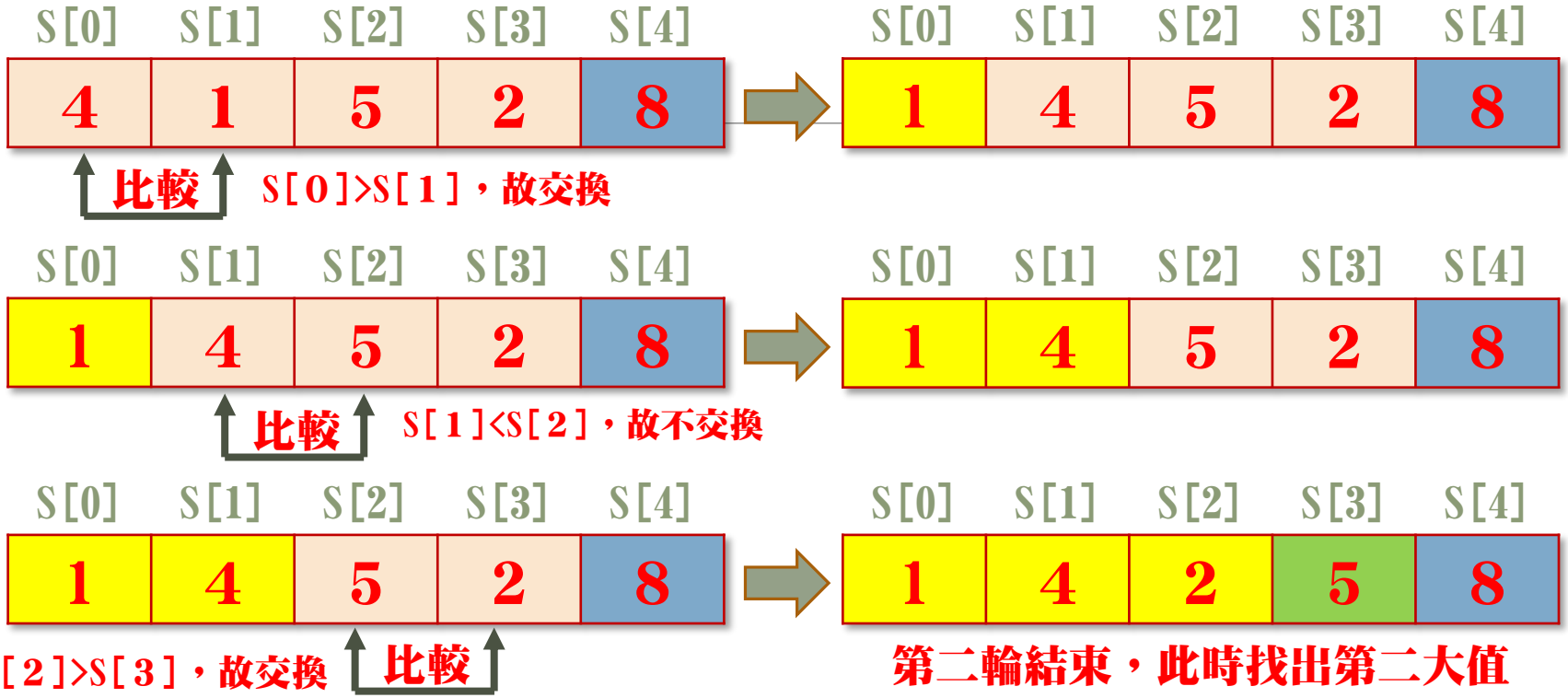


泡沫排序動作分析（由小到大為例）



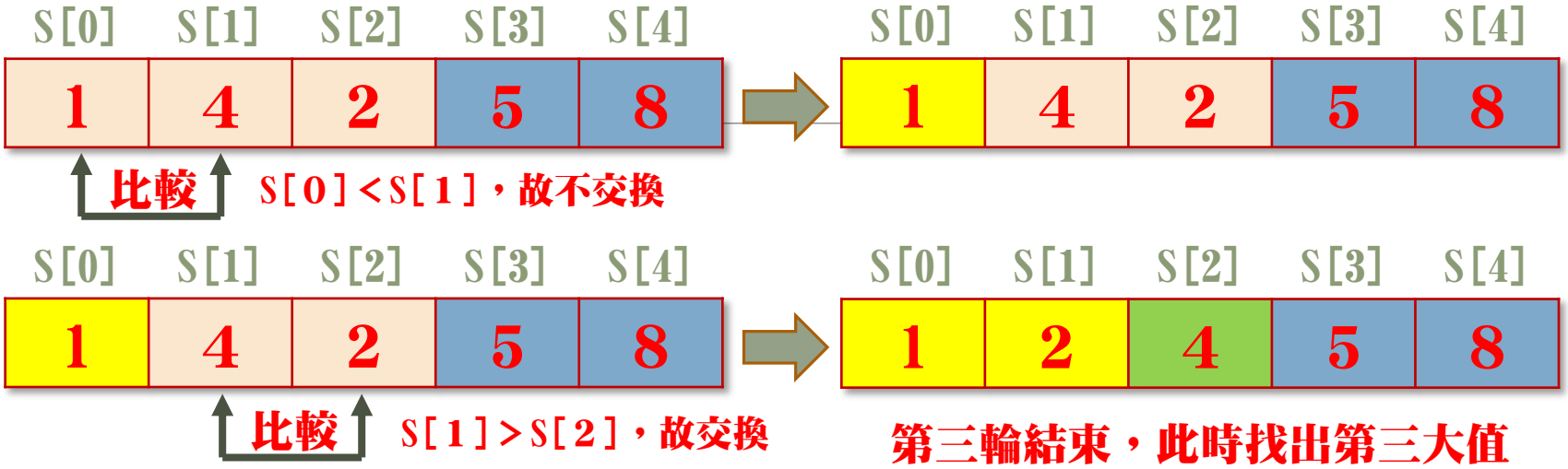
由以上動作可知，比較過程為：0、1 → 1、2 → 2、3 → 3、4

泡沫排序動作分析（由小到大為例）



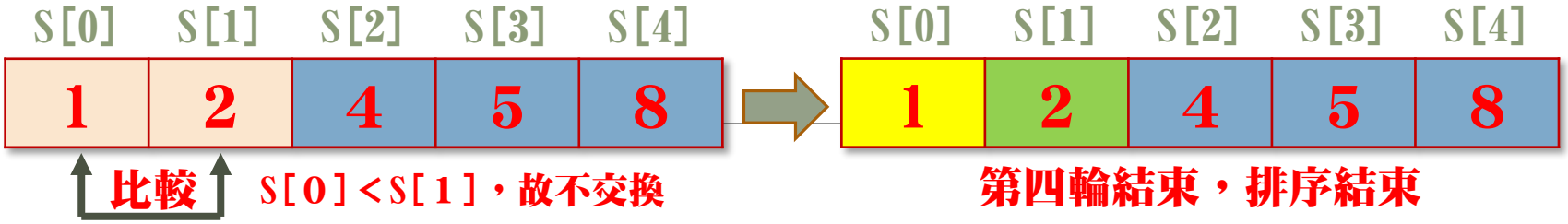
由以上動作可知，比較過程為：0、1 → 1、2 → 2、3

泡沫排序動作分析（由小到大為例）



由以上動作可知，比較過程為：0、1 → 1、2

泡沫排序動作分析（由小到大為例）



由以上動作可知，比較過程為：0、1

結論

第一輪比較過程為：0、1 → 1、2 → 2、3 → 3、4

第二輪比較過程為：0、1 → 1、2 → 2、3

第三輪比較過程為：0、1 → 1、2

第四輪比較過程為：0、1

由上可得演算法為：

```
for(i from 0 to length-1){
    for(j from 0 to length-1-i){
        if (array[j] > array[j+1])
            swap(array[j], array[j+1])
    }
}
```

陣列最大索引-1

泡沫排序實作

```
main(){
	int a[5]={3,4,2,9,8};
	int i,j,temp;

	for(i=0;i<=4-1;i++) {
		for(j=0;j<=4-1-I;j++){
			if(a[j]>a[j+1]){
				temp=a[j];
				a[j]=a[j+1];
				a[j+1]=temp;
			}
		}
	}
	cout<<"排序後的結果為：";
	for(i=0;i<5;i++){
		cout<<a[i]<<"  ";
	}
	cout<<endl;

	return 0;
}
```

二維陣列

二維陣列的結構如下圖，概念上是把多個一維陣列疊起來，存取時需要有行、列兩個索引值來索引。

陣列名稱[列][行]

列0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
列1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
	行0	行1	行2	行3	行4

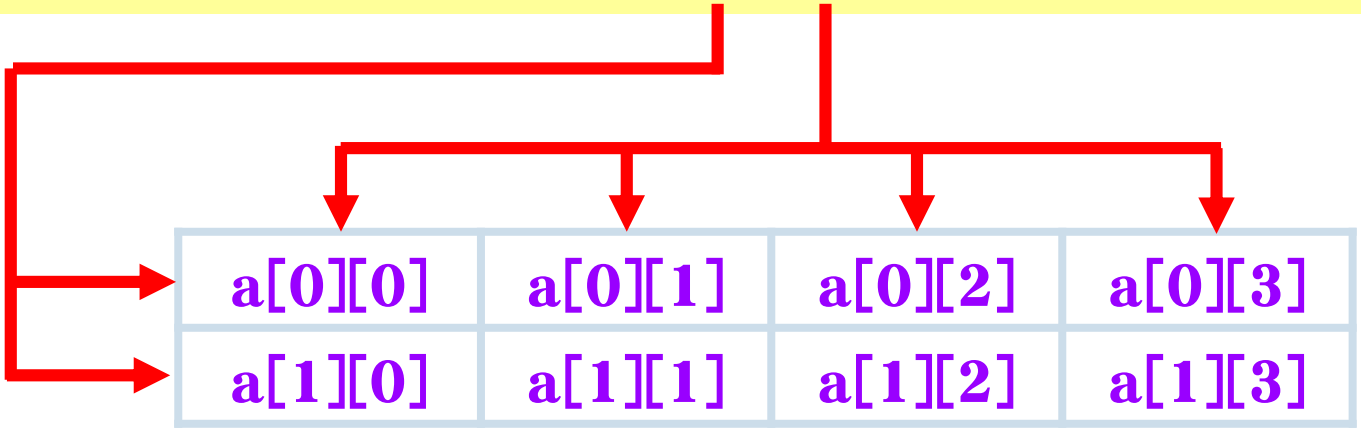
二維陣列宣告

- 語法格式(一)：不指定初值

```
資料型態 陣列名稱[列資料筆數][行資料筆數];
```

- 範例—

```
int a[2][4];
```



二維陣列宣告

- 語法格式(二)：指定初值

```
資料型態 陣列名稱[][行筆數]={{n1,n2..},
                              {n3,n4..},
                                .
                              {n6,n7..}};
```

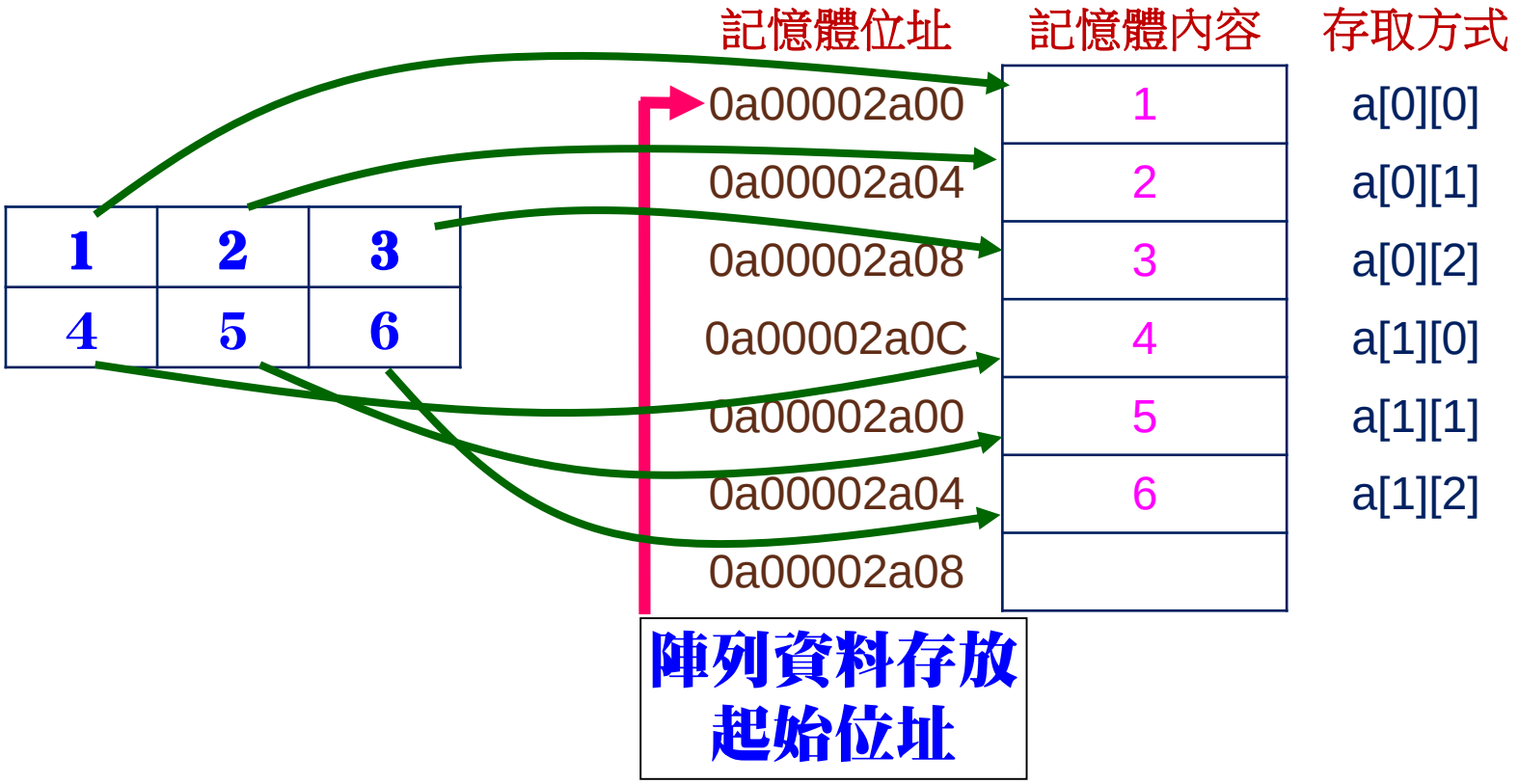
- 範例—

```
int a[][3]={{3,2,5},{6,9,9}};
```

一定要指定行筆數

二維陣列在記憶體中排列情形

以 int a[][3]={{1,2,3},{4,5,6}}為例



二維陣列資料的存取

和一維陣列一樣，可用迴圈來存取二維陣列。

將陣列中的資料列出

```
int a[2][5];

for(i=0;i<2;i++){
    for(j=0;j<5;j++){
        cout<<a[i][j];
    }
}
```

將資料輸入到陣列中

```
int a[2][5];

for(i=0;i<2;i++){
    for(j=0;j<5;j++){
        cin>>a[i][j];
    }
}
```

二維陣列範例—矩陣相加

數學中兩相同大小之矩陣可做相加的運算，其運算規則如下：

$$A=\begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad B=\begin{bmatrix} p & q \\ r & s \end{bmatrix}$$

則 $$A+B=\begin{bmatrix} a+p & b+q \\ c+r & d+s \end{bmatrix}$$

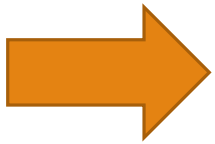
二維陣列範例—矩陣相加

```
main(){
        int m1[2][2]={{3,4},{2,2}};
        int m2[2][2]={{1,2},{3,7}};
        int m3[2][2];
        int i,j;

        cout<< "m1矩陣如下：\n";
        for(i=0;i<2;i++) {
                for(j=0;j<2;j++){
                        cout<<setw(3)<<m1[i][j];
                }
                cout<<"\n\n";
        }
        cout<< "m1矩陣如下：\n";
        for(i=0;i<2;i++) {
                for(j=0;j<2;j++){
                        cout<<setw(3)<<m2[i][j];
                }
                cout<<"\n\n";
        }
        cout<< "m1+m2結果為：\n";
        for(i=0;i<2;i++) {
                for(j=0;j<2;j++){
                  cout<<setw(3)<<m1[i][j]+m2[i][j];
                }
                cout<<endl;
        }
        return 0;
}
```

二維陣列範例—矩陣轉置

數學矩陣轉置的規則如下：

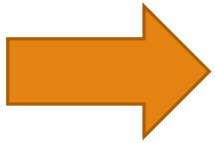
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$



轉置

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$$



轉置

$$\begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \end{bmatrix}$$

二維陣列範例—矩陣轉置

```
main(){
	char a[6][5]={{' ',' ','*',' ',' '},
	              {' ','*','*','*',' '},
	              {'*','*','*','*','*'},
	              {' ',' ','*',' ',' '},
	              {' ',' ','*',' ',' '},
	              {' ',' ','*',' ',' '}};
	char at[5][6];
	int i,j;
	cout<<"原始矩陣：\n\n";
	for(i=0;i<6;i++){
		for (j=0;j<5;j++){
			cout<<a[i][j];
			at[j][i]=a[i][j];
		}
		cout<<endl;
	}
	cout<<endl<<endl;
	cout<<"轉置後矩陣：\n\n";
	for(i=0;i<5;i++){
		for (j=0;j<6;j++){
			cout<<at[i][j];
		}
		cout<<endl;
	}


	return 0;
}
```

字串(一)

- 字串是由字元所形成的集合， C 語言的資料型態中沒有字串，但仍然可以經由字元陣列來組成字串的結構。
- 若要使用字串，可透過 char 關鍵字來宣告字元陣列，該字元陣列的名稱即可視為字串來使用。
- 把 字串儲存在字元陣列中， 還必須 在最尾端字元的後面再加上 '\0'（NULL）字元 當成字串的結束字元。

字串(二)

- 假如經由 單引號 來設定字元陣列的資料，則在字元資料的最後並不會加上結束字元 '\0' ；但若是以 雙引號 來設定字元陣列的初始值， 編譯器會 在結尾處自動加上 '\0' 。
- 在C++中可引用cstring 標頭檔，即可將變數宣告為 string 型態，字串資料需以 雙引號 包夾來設定資料，在字串結尾不會附加'\0'，使用時接近BASIC之字串型態；而此一標頭檔亦包含原來C語言之各字串函數。

字串—字元陣列

- 宣告範例1—以陣列的方式宣告

```
char s[]={'a','b','c','d','e','\0'};
```

在記憶體中的資料

a	b	c	d	e

- 宣告範例2—以字串的方式宣告(仍為陣列型態)

```
char s[]= "abcde";
```

在記憶體中的資料

a	b	c	d	e	\0

系統自己加上去的

字串範例一

將一字串內容逐字印出

```
main(){
        char s []="abcde";
        int i;

        i=0;

        while(s[i]!='\0') {
                cout<<s[i]<<endl;
                i++;
        }

        cout<<"字串長度為："<<i;
        return 0;
}
```

字串範例二

計算由數字所組成之字串中的數字總和。

```
main(){
        char s[]="123456789";
        int sum,i;

        i=0;sum=0;
        while(s[i]!='\0') {
                sum=sum+ (int)s[i]-(int)'0' ;
                i++;
        }

        cout<<"數值總和為："<<sum;
        return 0;
}
```

將字元轉為ASCII碼後減去0的ASCII碼，即為該數字文字的數值。

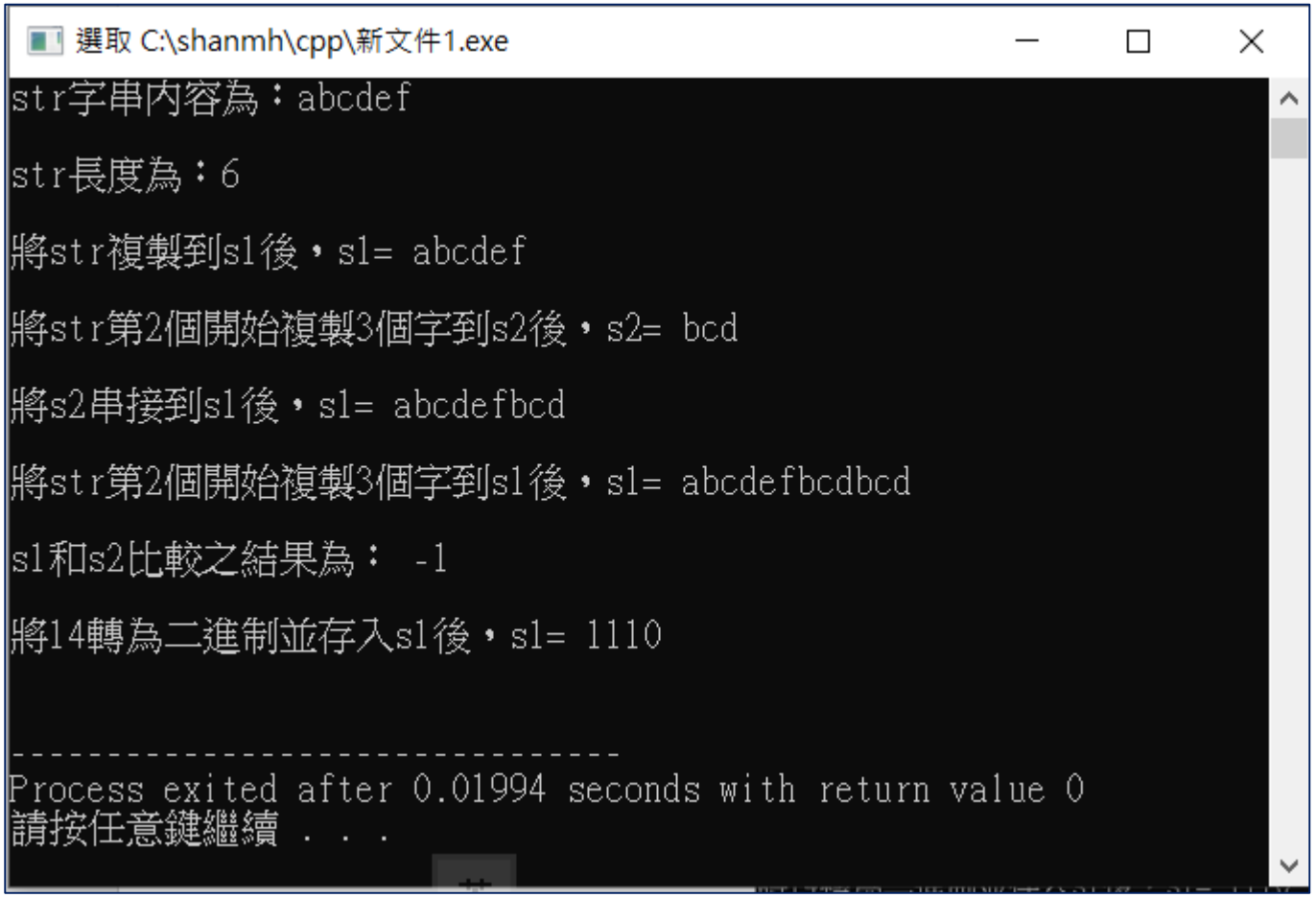
常用之字串函數(以字元組成字串)

函式	說明
strlen(char *str)	取得字串長度
strcpy(char *dest,char *src)	複製字串
strcpy(char *dest,char *src,size)	複製部份字串
strcat(char *dest,char *src)	字串串接
strncat(char *dest,char *src, size)	串接部份字串
strcmp(char *str1, char *str2)	字串比較　str1=str2 回傳0 str1>str2 回傳1 str1<str2 回傳-1
itoa(int val,char *str, int base)	將整數轉成字串(cstdlib)

字串函數範例(引用cstring、cstdlib)

```
main(){
	char str[]="abcdef";
	char s1[20],s2[20];

	cout<<"str字串內容為："<<str<<endl<<endl;
	cout<<"str長度為："<<strlen(str)<<endl<<endl;
	strcpy(s1,str);  cout<<"將str複製到s1後，s1= "<<s1<<endl<<endl;
	strncpy(s2,str+1,3);cout<<"將str第2個開始複製3個字到s2，s2= "<<s2<<endl<<endl;
	strcat(s1,s2);cout<<"將s2串接到s1後，s1= "<<s1<<endl<<endl;
	strncat(s1,str+1,3);cout<<"將str第2個開始複製3個字到s1，s1= "<<s1<<endl<<endl;
	cout<<"s1和s2比較之結果為： "<<strcmp(s1,s2)<<endl<<endl;
	itoa(14,s1,2);cout<<"將14轉為二進制並存入s1後，s1= "<<s1<<endl<<endl;

	return 0;
}
```

結果


選取 C:\shanmh\cpp\新文件1.exe
str字串內容為：abcdef
str長度為：6
將str複製到s1後，s1= abcdef
將str第2個開始複製3個字到s2後，s2= bcd
將s2串接到s1後，s1= abcdefbcd
將str第2個開始複製3個字到s1後，s1= abcdefbcdbcd
s1和s2比較之結果為： -1
將14轉為二進制並存入s1後，s1= 1110

Process exited after 0.01994 seconds with return value 0
請按任意鍵繼續 . . .

字串—引用cstring

- 引用後，可以 string 型態來宣告變數。
- 此型態之字串變數可以 + 執行字串串接，如:

  s1="1234" 、 s2="5678" 執 行 s1=s1+s2 後 ，s1="12345678"

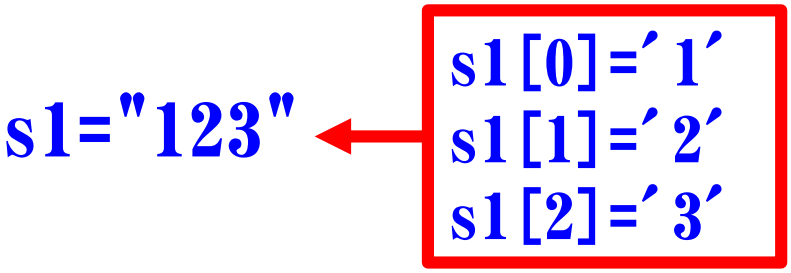
- 此型態之字串變數可以直接進行比較運算，如：s1="1234"、s2="5678"，執行：

  if(s1==s2) cout<<"相同";else cout<<"不同";

  結果為顯示 不同

字串—引用cstring

- 雖然為字串變數，但仍能以陣列的方式來存取。



- 若做字串資料輸入時，建議使用：

  getline(cin,字串變數);

  在輸入時可包含空格在內。

- 可使用 字串物件變數.length() 來得知字串長度

字串—引用cstring

- 宣告範例—以陣列的方式宣告

```
string s;
```
空字串

```
string s="";
```
空字串

```
string s="abc";
```
設定初值

字串反轉

將輸入之字串反轉後顯示出來(以string物件實作)

```
main(){
        string s1,s2;
        int n,i;

        cout<<"請輸入一字串:";getline(cin,s1);
        n=s1.length();
        s2="";
        for(i=n-1;i>=0;i--)
                s2=s2+s1[i];
        cout<<"反轉後之字串為："<<s2<<endl;
        if(s2==s1) cout<<“是迴文”; else cout<<“不是迴文”;

        return 0;
}
```