C++ 字串與結構

Su Huang

OUTLINE

1. 字元陣列
2. 字串
3. 結構
4. 列舉
5. 練習

OUTLINE

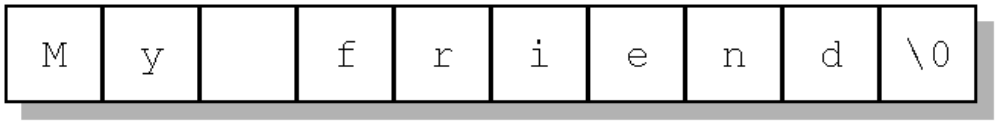
1. 字元陣列
2. 字串
3. 結構
4. 列舉
5. 練習

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-string 函數-1

- #include <cstring>

Display 9.1 Some Predefined C-String Functions in <cstring>

FUNCTION	DESCRIPTION	CAUTIONS
`strcpy(`*Target_String_Var*`,` *Src_String*`)`	Copies the C-string value *Src_String* into the C-string variable *Target_String_Var*.	Does not check to make sure *Target_String_Var* is large enough to hold the value *Src_String*.
`strcpy(`*Target_String_Var*`,` *Src_String*`,` *Limit*`)`	The same as the two-argument `strcpy` except that at most *Limit* characters are copied.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcpy`. Not implemented in all versions of C++.
`strcat(`*Target_String_Var*`,` *Src_String*`)`	Concatenates the C-string value *Src_String* onto the end of the C-string in the C-string variable *Target_String_Var*.	Does not check to see that *Target_String_Var* is large enough to hold the result of the concatenation.

(continued)

C-string 函數-2

Display 9.1 Some Predefined C-String Functions in <cstring>

FUNCTION	DESCRIPTION	CAUTIONS
`strcat(`*Target_String_Var*`,` *Src_String, Limit*`)`	The same as the two argument `strcat` except that at most *Limit* characters are appended.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcat`. Not implemented in all versions of C++.
`strlen(`*Src_String*`)`	Returns an integer equal to the length of *Src_String*. (The null character, `'\0'`, is not counted in the length.)	
`strcmp(`*String_1*`,` *String_2*`)`	Returns 0 if *String_1* and *String_2* are the same. Returns a value < 0 if *String_1* is less than *String_2*. Returns a value > 0 if *String_1* is greater than *String_2* (that is, returns a nonzero value if *String_1* and *String_2* are different). The order is lexicographic.	If *String_1* equals *String_2*, this function returns 0, which converts to `false`. Note that this is the reverse of what you might expect it to return when the strings are equal.
`strcmp(`*String_1*`,` *String_2, Limit*`)`	The same as the two-argument `strcat` except that at most *Limit* characters are compared.	If *Limit* is chosen carefully, this is safer than the two-argument version of `strcmp`. Not implemented in all versions of C++.

範例

```
#include <iostream>
#include <cstdlib>
#include <cstring>
using namespace std;
int main(void)
{
   char name[30];
   char how[15]=", how are you?";
   char name2[30];
   int i;
   for(i=0;i<2;i++)
   {
      cout << "What's your name? ";
      cin.getline(name, 15);
      strcat(name, how);
      strcpy(name2, name);
      cout << "Hi, " << name << endl << endl;
      cout << "使用strcpy: " << name2 << endl << endl;
   }
   system("pause");
   return 0;
}
```

- strcpy and strcat

OUTLINE

1. 字元陣列
2. 字串
3. 結構
4. 列舉
5. 練習

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 (可查用)

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 (append 及 substr)

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

Practice 1

- 寫一個程式，由鍵盤輸入兩個字串。

  寫一個函數，輸入為兩個字串，分別印出兩個字串的長度，並判斷此二字串是否相同。若相同，則保持原貌，印出字串，若不同，則將兩字串連在一起，並印出連在一起的字串。（字串的運算及函數）

注意：用 string 當做函數參數，僅傳內容（值）
若要傳遞位址，請用參照（或指標）

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];    也可用 copystr[i].assign(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

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

Practice（整合範例)

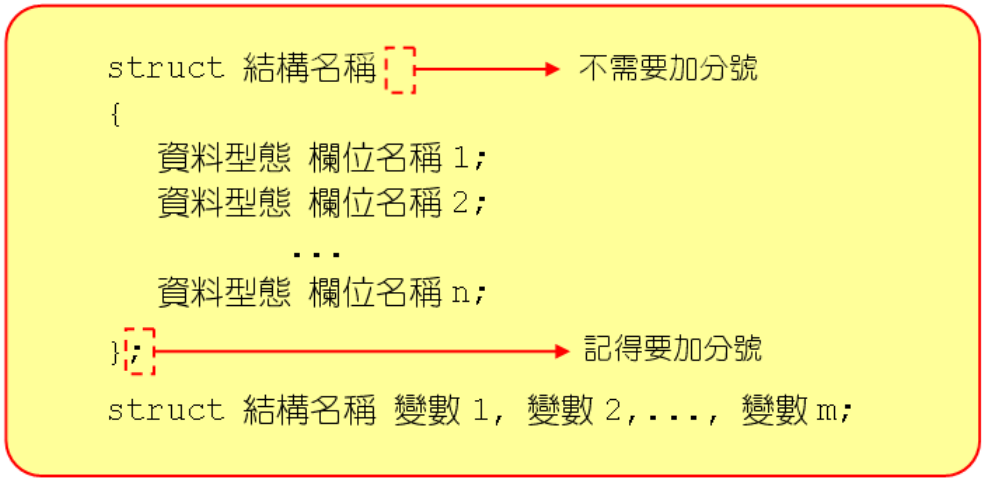
- 編碼應用<加密>
- 假設密碼表如下：
- {ABCDEFGHIJKLMNOPQRSTUVWXYZ}
- {DOJKZCTMYPAWHUQNBVGSRFXLIE}
- 輸入可以大寫或小寫
- 請寫一個函數，可處理編碼，先傳入輸入字串，然後轉換編碼後的字串，最後印出結果。

OUTLINE

1. 字元陣列
2. 字串
3. 結構
4. 列舉
5. 練習

結構的宣告 (1/2)

- 結構可以同時存放不同型態的資料於同一個結構體
- 結構的定義及宣告格式如下



- 下面是結構定義及宣告範例

```
struct mydata    // 定義結構 mydata
{
   string name;  // 各欄位的內容
   string id;
   int math;
   int eng;
};
```

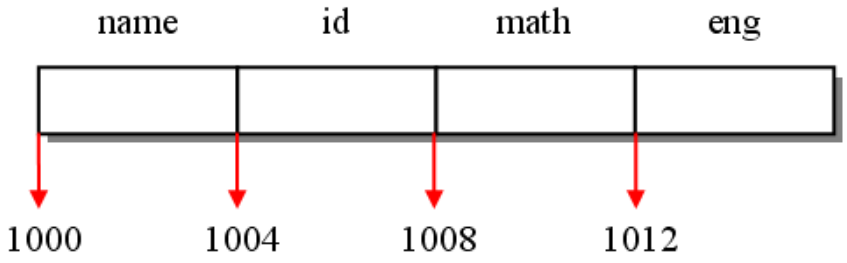
結構的宣告 (2/2)

- 您也可以用下列的格式來宣告結構

```
struct 結構名稱 [ ]  ──→ 不需要加分號
{
   資料型態 欄位名稱 1;
   資料型態 欄位名稱 2;
          ...
   資料型態 欄位名稱 n;
} 變數 1, 變數 2,..., 變數 m;
```

- 下面的結構定義及宣告範例為合法的格式：

```
struct mydata
{
   string name;
   string id;
   int math;
   int eng;
} student;
```



結構變數的使用及初值的設定

- 結構變數的使用格式

結構變數名稱 . 欄位名稱

- 結構內的成員可以利用小數點（.）來存取

結構變數的範例 (1/2)

- 下面的程式示範結構變數的輸入與輸出

```
01  // prog11_1, 結構變數的輸入與輸出
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  struct mydata                                   // 定義並宣告結構變數
07  {
08      string name;
09      int math;
10  } student;
11  int main(void)
12  {
13     cout << "Student's name:";
14     getline(cin,student.name);
15     cout << "Math score:";
16     cin >> student.math;
17     cout << "*****Output*****" << endl;          // 輸出結構變數內容
18     cout << student.name << "'s Math score is " << student.math;
19
20     system("pause");
21     return 0;
22  }
```

```
/* prog11_1 OUTPUT-------------
Student's name:Tippi Hong
Math score:95
*****Output*****
Tippi Hong's Math score is 95
-----------------------------*/
```

結構變數的範例 (2/2)

- 結構所佔用的記憶體有多少呢？請看看下面的程式

```
01  // prog11_2, 結構的大小
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  struct mydata                    // 定義結構
06  {
07     string name;
08     int math;
09  } student;
10  int main(void)
11  {
12     cout << "sizeof(student)=" << sizeof(student) << endl;
13
14     system("pause");
15     return 0;
16  }
```

字串僅儲存開始之記憶體位址
4 bytes

```
/* prog11_2 OUTPUT---
sizeof(student)=8
-------------------*/
```

設定結構變數的初值 (1/3)

- 用設定運算子（=）來設定結構變數的初值
- 宣告結構變數並為其設值的範例

```
struct mygood          // 定義結構 mygood
{
   string good;        // 貨品名稱
   int cost;           // 貨品成本
};
struct mygood first={"cracker",32};
```

- 在結構定義之後，直接宣告並設定變數的初值

```
struct mygood                // 定義結構 mygood
{
   string good;              // 貨品名稱
   int cost;                 // 貨品成本
} first={"cracker",32};      // 同時宣告變數 first,並設定初值
```


設定結構變數的初值 (2/3)

- 下面是設定結構變數初值的範例

```
01  // prog11_3, 結構變數的初值設定
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  struct mydata                                    // 定義並宣告結構變數
07  {
08     string name;
09     int math;
10  } student={"Mary Wang",74};                      // 設定結構變數初值
11  int main(void)
12  {
13     cout << "Student's name:" << student.name;  // 輸出結構變數內容
14     cout << endl << "m ath score=" << student.math << endl;
15
16     system("pause");
17     return 0;
18  }
```

```
/* prog11_3 OUTPUT-------

Student's name:Mary Wang
Math score=74

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

設定結構變數的初值 (3/3)

- 將結構變數**x**設給另一個結構變數**y**的練習

```
01  // prog11_4, 結構的設值
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  struct mydata                    // 定義結構
07  {
08     string name;
09     int age;
10  } x;                             // 宣告結構變數
11  int main(void)
12  {
13     struct mydata y={"Lily Chen",18};
14     x=y;
15     //輸出結構變數內容
16     cout << "x.name=" << x.name << ", x.age=" << x.age << endl;
17     cout << "y.name=" << y.name << ", y.age=" << y.age << endl;
18
19     system("pause");
20     return 0;
21  }
```

```
/* prog11_4 OUTPUT---------
x.name=Lily Chen, x.age=18
y.name=Lily Chen, y.age=18
--------------------------*/
```

Practice 2

- 請寫一個程式，使用結構的方式。由鍵盤輸入學生資料，其項目包含學號、姓名、期中考成績、期末考成績及平時成績，其學期成績則是以期中、期末佔 30%，平時成績佔 40% 計算。輸出項目除了該生的資料外，還要顯示學期成績。

Practice 2

- 建構一個結構陣列包含以下資訊：
  - 學生姓名 string
  - 學號 string
  - 期中成績 int
  - 期末成績 int
  - 學期總成績 double
- 於主程式中以鍵盤輸入學生人數、以及所有學生資訊（不含學期總成績）。
- 寫一個函數，將整個結構陣列輸入，計算出每位學生之學期總成績=期中成績＊0.4+期末成績*0.6。
- 於主程式中印出學生姓名及學期總成績。

OUTLINE

1. 字元陣列
2. 字串
3. 結構
4. 列舉
5. 練習

將整個結構傳遞到函數 (1/3)

- 下面列出將結構傳遞到函數中的格式

```
struct 結構名稱 1
{
   資料型態 欄位名稱 1;
        ...
   資料型態 欄位名稱 n;
} 變數 1, 變數 2,..., 變數 m;

傳回值型態 函數名稱(struct 結構名稱 1 變數名稱 1);      // 函數原型

int main(void)
{
        ...
   函數名稱(結構變數名稱);
}

傳回值型態 函數名稱(struct 結構名稱 1 變數名稱 1)
{       ...       }
```

將整個結構傳遞到函數 (2/3)

- 下面的程式，是將結構變數當成引數傳入函數中

若要傳遞結構的位址，需用參照方式

```
01  // prog11_5, 結構與函數
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  struct mydata
07  {
08     string name;
09     int age;
10  };
11  void func(struct mydata);
12  int main(void)
13  {
14     struct mydata woman={"Mary Wu",5};      // 宣告結構變數
15     cout << "before process..." << endl;
16     cout << "In main(), " << woman.name;     // 印出結構變數內容
17     cout << "'s age is " << woman.age << endl;
18     cout << "after process..." << endl;
19     func(woman);                             // 呼叫 func()函數
20     cout << "In main(), " << woman.name;
21     cout << "'s age is " << woman.age << endl;
```

```
/* prog11_5 OUTPUT --------------

before process...
In main(), Mary Wu's age is 5
after process...
In func(), Mary Wu's age is 15
In main(), Mary Wu's age is 5

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

使用結構名稱呼叫函數僅傳值

將整個結構傳遞到函數 (3/3)

```
22
23      system("pause");
24      return 0;
25  }
26
27  void func(struct mydata a)                    // 自訂函數 func()
28  {
29      a.age+=10;
30      cout << "In func(), " << a.name;          // 印出結構變數內容
31      cout << "'s age is " << a.age << endl;
32      return;
33  }
                    /* prog11_5 OUTPUT --------------

                    before process...
                    In main(), Mary Wu's age is 5
                    after process...
                    In func(), Mary Wu's age is 15
                    In main(), Mary Wu's age is 5

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

此時 a 表示主程式中的 woman

將結構欄位分別傳遞

- 程式prog11_6僅將結構變數的部分欄位傳入函數

```
01  // prog11_6, 將結構欄位分別傳遞到函數
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  struct mydata                          // 定義結構
07  {
08     string name;
09     int math;
10     int eng;
11  };
12  float avg(int,int);                    // 函數原型
13  int main(void)
14  {
15     struct mydata num={"Alice",71,80};          // 宣告結構變數
16     cout << num.name << "'s Math score=" << num.math;// 印出結構變數內容
17     cout << endl << "English score=" << num.eng << endl;
18     cout << "average=" << avg(num.math,num.eng) << endl;
19
20     system("pause");
21     return 0;
22  }
23
24  float avg(int a,int b)                 // 自訂函數 avg()
25  {
26     return (float)(a+b)/2;
27  }
```

函數僅傳值

```
/* prog11_6 OUTPUT ------
Alice's Math score=71
English score=80
average=75.5
----------------------*/
```

傳遞結構的位址 (1/2)

- 下面的程式利用指標的方式傳遞結構變數到函數

```
01  // prog11_7, 以指標傳遞結構到函數
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  struct data                                    // 定義結構
07  {
08     string name;
09     int a,b;
10  };
11  void change(struct data *),prnstr(struct data);    // 函數原型
12  int main(void)
13  {
14     struct data first={"David Young",9,2};          // 宣告結構變數
15     prnstr(first);
16     cout << "after process..." << endl;
17     change(&first);
18     prnstr(first);
19
20     system("pause");
21     return 0;
22  }
```

注意指標與參考的函數呼叫

```
/* prog11_7 OUTPUT ----
name=David Young
a=9     b=2
after process...
name=David Young
a=2     b=9

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

傳遞結構的位址 (2/2)

```
23
24  void change(struct data *ptr)          // 自訂函數 change()
25  {
26     int temp;
27     temp=ptr->a;              // ptr->a 可取出 ptr 所指向之結構的欄位 a 之值
28     ptr->a=ptr->b;            // 取出欄位 b 的值，並設定給欄位 a 存放
29     ptr->b=temp;              // 將 temp 設定給 ptr 所指向之結構的欄位 b 存放
30     return;
31  }
32
33  void prnstr(struct data in)            // 印出結構變數內容
34  {
35     cout << "name=" << in.name << endl;
36     cout << "a=" << in.a << "\t";
37     cout << "b=" << in.b << endl;
38     return;
39  }
```

以指標方式傳遞，需將 . 改為 ->

```
/* prog11_7 OUTPUT ----
name=David Young
a=9     b=2
after process...
name=David Young
a=2     b=9
--------------------*/
```

結構陣列

- 陣列的每一個成員都具有相同的結構型態稱之。
- 例：
- 假設一個公司裡有多個員工，每個員工都有姓名以及電話等資料。
- 見下頁範例

StructArray.cpp (1)

```
#include <iostream>
using namespace std;

const int NameSize  = 20;
const int PhoneSize = 10;

struct Employee
{
        char  Name[NameSize];
        char  Phone[PhoneSize];
};
```

StructArray.cpp (2)

```
int main()
{
        const int Size = 2;
        Employee Officer[Size];
        cout << "共 " << Size << " 個 Officers:\n";

        for (int i=0; i<Size; i++)
        {
                cout << "請輸入 Officer[" << i
                        << "] 的姓名: ";
                cin.getline(Officer[i].Name, NameSize, '\n');
                cout << "電話號碼: ";
                cin.getline(Officer[i].Phone,PhoneSize, '\n');
        }
```

StructArray.cpp (3)

```
for (int i=0; i<Size; i++)
        {
                cout << "Officer["   << i << "] 的資料是:\n"
                        << "姓名    : " << Officer[i].Name  << '\n'
                        << "電話號碼: " << Officer[i].Phone << '\n';
        }

        system("pause");
        return 0;
}
```

結構陣列於函數中傳遞

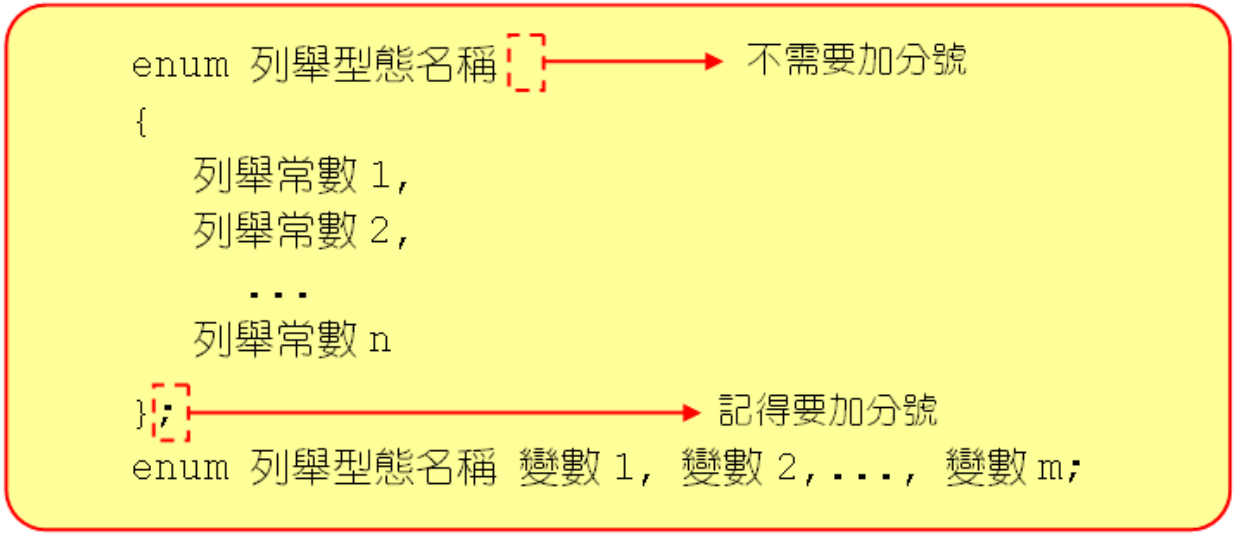
- 方式與陣列傳遞相似！！
- 陣列名稱代表該陣列之開始位址
- 練習：
  - 請寫一個函數，可將上述之範例中的兩個 Officer 資料對調，於主程式中印出結果。

```
void change(Employee stuff[])
{
    Employee temp;

    temp=stuff[0];
    stuff[0]=stuff[1];
    stuff[1]=temp;

    }
```

```
void change(Employee []);  //函數原形

change(Officer);        //函數呼叫
```

列舉型態的定義及宣告 (1/2)

- 列舉型態可以定義某種資料型態，並設定此資料型態內所包含的成員，以方便程式碼的撰寫
- 列舉型態的定義及宣告格式如下



列舉型態的定義及宣告 (2/2)

- 下面為列舉型態定義及宣告的範例

```
enum desktop                    // 定義列舉型態 desktop
{pen,pencil,eraser,book,tape};
enum desktop mine;              // 宣告列舉型態 desktop 之變數 mine
```

- 列舉型態的定義及宣告格式之另一種方式

```
enum desktop              // 定義列舉型態 desktop
{   pen,pencil,eraser,
    book,tape
} mine;                   // 宣告列舉型態 desktop 之變數 mine
```

列舉型態的使用與設值 (1/8)

- 列舉型態會自動轉換成整數型態
- 整數型態不會自動轉換成為列舉型態
- 整數與列舉型態的轉換格式

```
列舉型態變數=static_cast<列舉型態名稱>(欲轉換之內容);
```

- 下面列出合法與不合法的設值方式

```
May;                                  // 合法的列舉型態變數設值
six=static cast<month>(3);            // 合法的列舉型態變數設值
six=static cast<month>(six+2);        // 合法的列舉型態變數設值
six=4;                                // 不合法的列舉型態變數設值
six="May";                            // 不合法的列舉型態變數設值
six=July;                             // 不合法的列舉型態變數設值
```

列舉型態的使用與設值 (2/8)

- 下面的程式示範列舉型態變數的使用方式

```
01  // prog11_12, 列舉型態的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  enum month  // 定義列舉型態
06  {  January,February,March,
07     April,May,June } six;
08  int main(void)
09  {
10     cout << "sizeof(six)=" << sizeof(six) << endl; // 列舉型態的長度
11     cout << "January=" << January << endl;       // 印出列舉常數的值
12     cout << "February=" << February << endl;
13     cout << "March=" << March << endl;
14     cout << "April=" << April << endl;
15     cout << "May=" << May << endl;
16     cout << "June=" << June << endl;
17
18     system("pause");
19     return 0;
20  }
```

```
/* prog11_12 OUTPUT---
sizeof(six)=4
January=0
February=1
March=2
April=3
May=4
June=5
--------------------*/
```

列舉型態的使用與設值 (3/8)

- 列舉常數值會由所設定的值開始遞增，如下面的程式

```
01  // prog11_13, 列舉常數的設值
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  enum month  // 定義列舉型態
06  {  January,February,March=4,                    // 將 March 設值為 4
07     April,May,June } six;
08  int main(void)
09  {
10     cout << "January=" << January << endl;   // 印出列舉常數的值
11     cout << "February=" << February << endl;
12     cout << "March=" << March << endl;
13     cout << "April=" << April << endl;
14     cout << "May=" << May << endl;
15     cout << "June=" << June << endl;
16
17     system("pause");
18     return 0;
19  }
```

```
/* prog11_13 OUTPUT------
January=0
February=1
March=4
April=5
May=6
June=7
----------------------*/
```

列舉常數值因設定而隨之更改

列舉型態的使用與設值 (4/8)

- 下面的程式是將列舉型態中的列舉常數印出

```
01  // prog11_14, 列舉型態的使用
02  #include <iostream>
03  #include <cstdlib>
04  #include <string>
05  using namespace std;
06  enum month                          // 定義列舉型態
07  {  January,February,March,
08     April,May,June } six;
09  int main(void)
10  {
11     string a[6]={"January","February","March",
12                  "April","May","June"};
13     for(six=January;six<=June;six=static cast<month>(six+1))
14        cout << "six(" << six << ")=" << a[six] << endl;
15
16     system("pause");
17     return 0;
18  }
```

```
/* prog11_14 OUTPUT----

six(0)=January
six(1)=February
six(2)=March
six(3)=April
six(4)=May
six(5)=June

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

列舉型態的使用與設值 (5/8)

- 下面的程式是利用列舉型態模擬滑鼠的三個按鈕

```
01  // prog11_15, 列舉型態的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     enum mykey                                  // 定義列舉型態
08     {
09        left,right,middle
10     } mouse;                                    // 宣告列舉型態變數
11     int key;
12     do                                          // 輸入 0~2 的值
13     {
14        cout << "Button press?(0)Left (1)Right (2)Middle: ";
15        cin >> key;
16     } while((key>2)||(key<0));
17     mouse=static cast<mykey>(key);
```

```
/* prog11_15 OUTPUT ------------------------
Button press?(0)Left (1)Right (2)Middle: 5
Button press?(0)Left (1)Right (2)Middle: 2
Middle Button Pressed!
-----------------------------------------*/
```

列舉型態的使用與設值 (6/8)

```
18      switch(mouse)                          // 根據 key 的值印出字串
19      {
20         case left:  cout << "Left Button Pressed!" << endl;
21                     break;
22         case right: cout << "Right Button Pressed!" << endl;
23                     break;
24         case middle:cout << "Middle Button Pressed!" << endl;
25      }
26
27      system("pause");
28      return 0;
29  }
```

```
/* prog11_15 OUTPUT ------------------------

Button press?(0)Left (1)Right (2)Middle: 5
Button press?(0)Left (1)Right (2)Middle: 2
Middle Button Pressed!

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

列舉型態的使用與設值 (7/8)

- 設定列舉變數初值的範例

```
enum sports      // 定義列舉型態 sports
{
   tennis,swimming,baseball,ski
} favorite=ski;  // 設定 favorite 的初值為 ski
```

- 下面的程式是為列舉型態變數設值的範例

```
01  // prog11_16, 列舉變數的設值
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  enum sports                                  // 定義列舉型態
06  {
07     tennis,swimming,baseball,ski
08  } favorite=ski;                              // 宣告列舉變數並設值
09  int main(void)
10  {
```

```
/* prog11_16 OUTPUT---
favorite=ski
---------------------*/
```

列舉型態的使用與設值 (8/8)

```
11      cout << "favorite=";                    // 印出列舉變數所對應的內容
12      switch(favorite)
13      {
14         case 0:cout << "tennis" << endl;
15                break;
16         case 1:cout << "swimming" << endl;
17                break;
18         case 2:cout << "baseball" << endl;
19                break;
20         case 3:cout << "ski" << endl;
21      }
22
23      system("pause");
24      return 0;
25  }
```

```
/* prog11_16 OUTPUT---
favorite=ski
---------------------*/
```

typedef

- typedef是type definition的縮寫，就是定義型態之意
- typedef的使用格式如下所示

```
typedef 資料型態 識別字;
```

- 下面的型態定義及宣告範例

```
typedef int clock;    // 定義 clock 為整數型態
clock hour,second;    // 宣告 hour,second 為 clock 型態
```

使用 typedef

- 程式prog11_17是利用typedef自訂資料型態的範例

```
01  // prog11_17, 自訂型態—typedef 的使用
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     typedef float temper;                    // 定義自訂型態
08     temper f,c;                              // 宣告自訂型態變數
09     cout << "Input Celsius degrees:";
10     cin >> c;
11     f=(float)(9.0/5.0)*c+32;                 // 轉換公式
12     cout << c << " Celsius is equal to ";    // 印出轉換後的結果
13     cout << f << " Fahrenheit degrees" << endl;
14
15     system("pause");
16     return 0;
17  }
```

#define與typedef

- 在某些情況下，#define可以取代typedef

```
typedef int clock;    // 定義 clock 為整數型態
clock hour,second;    // 宣告 hour,second 為 clock 型態
```

- 在此可將 #define取代為typedef，成為如下面的敘述

```
#define CLOCK int     // 定義識別名稱 CLOCK 為 int
CLOCK hour,second;    // 前置處理器會將 CLOCK 替換為 int
```

- 使用typedef時是由編譯器來執行
- #define是由前置處理器主導

typedef的使用範例 (1/2)

- 下面的程式是使用typedef自訂新的型態之範例

```
01  // prog11_18, 自訂型態—typedef 的使用
02  #include <iostream>
03  #include <cstdlib>
04  #include <iomanip>                    // 將標頭檔 iomanip 含括進來
05  using namespace std;
06  typedef struct                        // 定義自訂型態
07  {
08     int hour;
09     int minite;
10     float second;
11  } mytime;
12  void subs(mytime t[]);                // 函數原型
13  int main(void)
14  {
15     int i;
16     mytime t[3]={{6,24,45.58f},{3,40,17.43f}};
17     cout << setfill('0');
18     subs(t);                           // 呼叫subs()函數,計算t[0]+t[1]
19     for(i=0;i<3;i++)                   // 印出陣列內容
20     {
21        cout << "t[" << i << "]=" << setw(2) << t[i].hour << ":";
22        cout << setw(2) << t[i].minite << ":";
23        cout << setw(5) << t[i].second << endl;
24     }
```

```
/* prog11_18 OUTPUT----

t[0]=06:24:45.58
t[1]=03:40:17.43
t[2]=10:05:03.01

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

typedef的使用範例 (2/2)

```
25
26      system("pause");
27      return 0;
28  }
29
30  void subs(mytime t[])                    // 自訂函數 subs()
31  {
32      int count2=0,count3=0;
33      t[2].second=t[0].second+t[1].second;                    // 秒數相加
34      while(t[2].second>=60)
35      {
36          t[2].second-=60;
37          count3++;
38      }
39      t[2].minite=t[0].minite+t[1].minite+count3;             // 分數相加
40      while(t[2].minite>=60)
41      {
42          t[2].minite-=60;
43          count2++;
44      }
45      t[2].hour=t[0].hour+t[1].hour+count2;                   // 時數相加
46      return;
47  }
```

```
/* prog11_18 OUTPUT----
t[0]=06:24:45.58
t[1]=03:40:17.43
t[2]=10:05:03.01
----------------------*/
```

OUTLINE

1. 字元陣列
2. 字串
3. 結構
4. 列舉
5. 練習

Practice1

- 請找出名為 BILL 的人是在哪一行的第幾個(從1開始)，最多隻會有一位 BILL，而如果 BILL 不在這裡面，則輸出 NO。

INPUT：

輸入資料共有三行文字，每行有一個以上的名字，代表這行隊伍裏位置的順序。

OUTPUT：

請找出名為 BILL 的人是在哪一行的第幾個(從1開始)，最多隻會有一位 BILL，而如果 BILL 不在這裡面，則輸出 NO。

範例輸入：

A BILL C
D E F G
HI

範例輸出 ：

1 2

格式要求

int *func(string[3]);

Practice 2

- 寫一個程式，由鍵盤輸入兩個字串。

  寫一個函數，輸入為兩個字串，分別印出兩個字串的長度，並判斷此二字串是否相同。若相同，則保持原貌，印出字串，若不同，則將兩字串連在一起，並印出連在一起的字串。（字串的運算及函數）

INPUT：
兩個String

OUTPUT：

若兩字串相同 則印出此String，若不同則輸出兩字串拚接之String。

範例輸入：

StevenLiuStevenLiu

EasonHuang ZHEHUI

範例輸出 ：

StevenLiu

EasonHuangZHEHUI

格式要求

```
string func1(string,string);
int main(){
        string = func1(string,string)
        cout << string;
}
```

Practice 3

- 建構一個結構陣列包含以下資訊：
  - 學號 string
  - 期中成績 int
  - 期末成績 int
  - 學期總成績 double
- 於主程式中以鍵盤輸入所有學生資訊（不含學期總成績）。
- 寫一個函數，將整個結構陣列輸入，計算出每位學生之學期總成績=期中成績＊0.4+期末成績*0.6。
- 於主程式中印出學號與學期總成績。

INPUT：
所有學生資訊
OUTPUT：
學期總成績

範例輸入：
0711999 80 90
範例輸出 ：
0711087 86

格式要求

```
typedef struct student
{
        …
}student;
Int func(student)
{
}
```