

## 第十一章 結構、聯合與列舉

# 第十一章 大綱

---

- ▶ 11-1 結構
  - ▶ 結構的定義
  - ▶ 結構變數的宣告
  - ▶ 結構陣列的宣告
- ▶ 11-2 函式呼叫的參數傳遞與值的傳回
  - ▶ 將結構傳遞給被呼叫函式
  - ▶ 將結構當成傳返回值傳回給呼叫函式
- ▶ 11-3 結構與指標
  - ▶ 定義指向結構的指標
  - ▶ 透過指標存取結構成員
- ▶ 11-4 聯合(略)
- ▶ 11-5 列舉
- ▶ 11-6 typedef指令
- ▶ 11-7 程式練習(略)

## 11-1 結構：概述結構

---

- ▶ 第二章曾經談過C的**基本資料型態**(又稱**系統內定的資料型態**)有：
  - ▶ int, short, long (儲存整數資料)
  - ▶ float, double (儲存浮點數資料)
  - ▶ char (字元)
- ▶ 結構是**使用者自訂的資料型態**，結構是多個資料欄位所形成的集合，結構中的資料欄位有以下形式：
  - ▶ 前述的基本資料型態
  - ▶ 陣列
  - ▶ 結構(巢狀結構)

## 11-1 結構：宣告語法

---

- ▶ 其宣告的語法如下：

```
struct 結構名稱 {  
    結構成員1;  
    結構成員2;  
    ...  
} 變數名稱;
```

- ▶ 以上是結構定義與變數宣告寫在一起
- ▶ struct 是關鍵字
- ▶ 結構名稱及變數名稱是使用者自訂名稱

## 11-1 結構：範例

---

- ▶ 結構宣告與變數宣告寫在一起：

```
struct animal {  
    char name[10];  
    int sex;  
}dog;
```

- ▶ 上方的範例中，定義了一個名稱為animal的結構，結構包含了兩個成員，第1個成員為字元陣列name，第2個成員為整數變數sex。
- ▶ 以struct animal為型態名稱，定義一個名稱為dog的變數

- ▶ 結構宣告與變數宣告分開寫：

```
struct animal {  
    char name[10];  
    int sex;  
};  
struct animal dog;
```

- ▶ 在以上變數宣告中，
  - ▶ struct animal 是型態名稱
  - ▶ dog是變數名稱

## 11-1 結構：記憶體配置

---

- ▶ 系統會為每一個結構成員各配置記憶體空間
- ▶ 以下列結構為例，在記憶體中的配置情形示意圖如下

```
struct animal {  
    char name[10];  
    int sex;  
}dog;
```

name[10] (10bytes)	sex(4bytes)
--------------------	-------------

## 11-1 結構：練習題

---

- ▶ 定義一結構 及宣告結構變數：
  - ▶ 結構名稱：StudentScore
  - ▶ 結構成員的型態及名稱：
    - ▶ 學號：char id[9];
    - ▶ 數學成績：int math;
    - ▶ 英文成績：int english;
    - ▶ 國文成績：int chinese;
- ▶ 宣告一結構變數，變數名稱為s1;
- ▶ Q1：將結構定義與變數宣告寫在一起
- ▶ Q2：將結構定義與變數宣告分開寫

## 11-1 結構：結構成員的存取

---

```
struct animal {  
    char name[10];  
    int sex;  
};
```

```
struct animal dog, cat;
```

▶ 我們使用『.』運算子來存取結構成員，以上方宣告為例，存取sex成員的語法如下：

```
dog.sex = 1;    //假設1為公  
cat.sex = 0;    //假設0為母
```

▶ 若要利用gets( )函式輸入一隻公狗及一隻母貓的姓名資料，語法如下：

```
gets(dog.name);  
gets(cat.name);
```

▶ 相關程式碼：  
Struct\_MemberAccess.c



## 11-1 結構：程式練習題(結構成員的存取)



```
C:\D:\C\Chap11\chap11_Example\Struct_MemberAcce...
請輸入狗的性別：<0:母,1:公>1
請輸入貓的性別：<0:母,1:公>0
請輸入狗的名字：John
請輸入貓的名字：Mary

已輸入的動物清單：
-----:
性別<0:母,1:公>      名字
      1              John
      0              Mary
請按任意鍵繼續 . . .
```

- ▶ 開啟 Struct\_MemberAccess.c，在Line 27以後加入輸出指令，以產生左列的程式輸出結果：
- ▶ 提示：
  - ▶ fflush(stdin);
    - ▶ 清除殘留在標準輸入設備中的資料
  - ▶ 輸出格式：
    - ▶ tab
    - ▶ 性別
    - ▶ 兩個tab
    - ▶ 名字

## 11-1 結構：結構陣列的宣告與存取

---

- ▶ 結構是一種使用者自訂的資料型態，和基本資料型態(也稱系統內定的資料型態)一樣，均可用於宣告陣列
- ▶ 宣告語法  
型態 名稱[長度];
- ▶ 範例  

```
struct animal dog[10];
```
- ▶ 存取語法：陣列名稱[索引].成員名稱;
- ▶ 範例：  

```
dog[3].sex = 0; //指定第3隻狗的性別為母  
scanf("%d", &cat[1].sex); //請使用者輸入第1隻貓的性別
```

## 11-1 結構：結構陣列的練習題

---

```
struct animal {  
    char name[10];  
    int sex;  
};
```

► Q3：

以上述結構宣告結構陣列，陣列名稱為dog，陣列維度為5

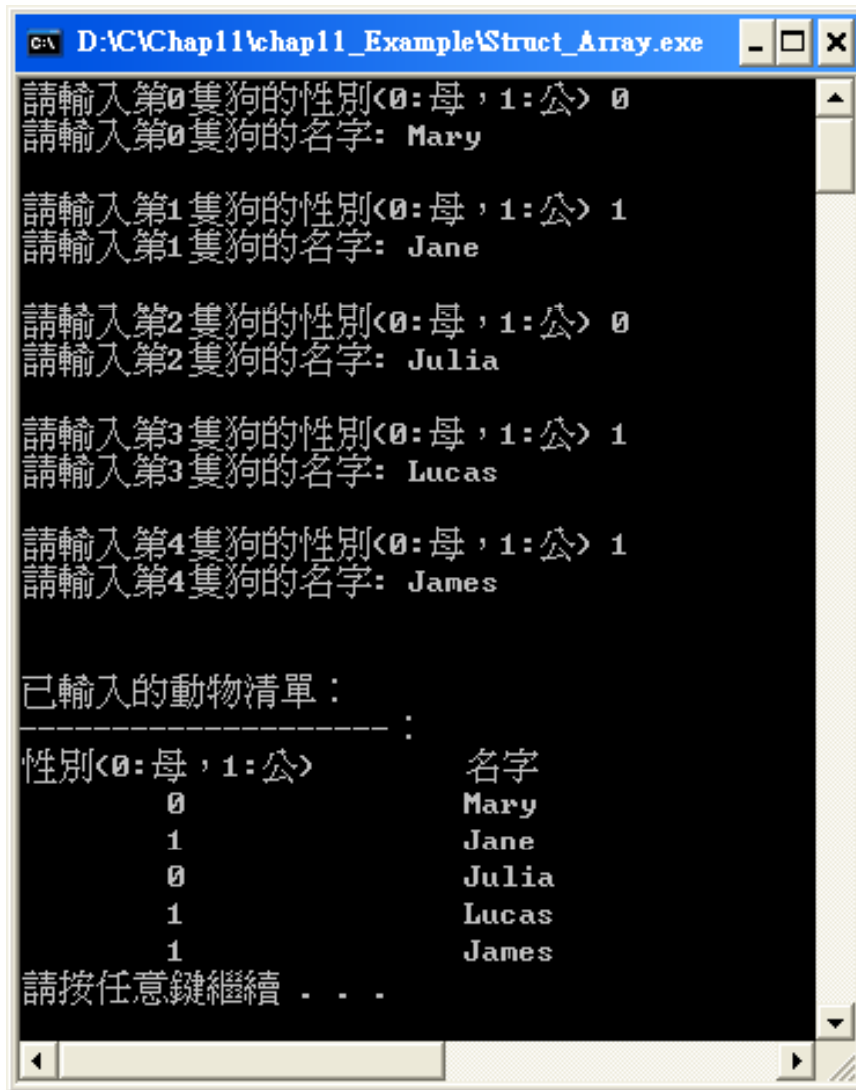
► Q4：第0隻狗的名字如何存取？

► Ans:

► Q5：第3隻狗的性別如何存取？

► Ans:

## 11-1 結構：程式練習題(結構陣列的存取)



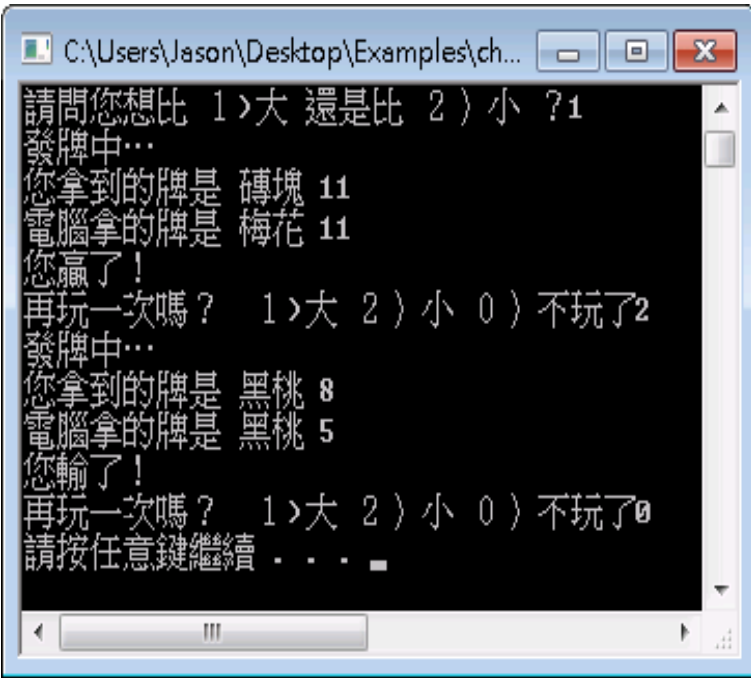
```
C:\D:\C\Chap11\chap11_Example\Struct_Array.exe
請輸入第0隻狗的性別<0:母,1:公> 0
請輸入第0隻狗的名字: Mary
請輸入第1隻狗的性別<0:母,1:公> 1
請輸入第1隻狗的名字: Jane
請輸入第2隻狗的性別<0:母,1:公> 0
請輸入第2隻狗的名字: Julia
請輸入第3隻狗的性別<0:母,1:公> 1
請輸入第3隻狗的名字: Lucas
請輸入第4隻狗的性別<0:母,1:公> 1
請輸入第4隻狗的名字: James

已輸入的動物清單:
-----:
性別<0:母,1:公>      名字
      0              Mary
      1              Jane
      0              Julia
      1              Lucas
      1              James
請按任意鍵繼續 . . .
```

- ▶ 開啟Struct\_Array.c，完成以下程式功能：
  - ▶ 宣告結構陣列
  - ▶ 以迴圈從標準輸入設備讀取資料並儲存於結構陣列中
  - ▶ 以迴圈輸出結構陣列的內容
- ▶ 程式執行結果如左列視窗

## 程式範例：玩牌比大小程式

- ▶ 學習重點：結構的使用
- ▶ 【參考檔案】11-1-1.c
- ▶ 程式設計目標：請撰寫一個撲克牌的遊戲，隨機各發一張牌給電腦與使用者，使用者可選擇要比大或比小，之後輸出誰獲勝，並且設計離開程式的選項。程式的操作畫面如下圖所示。



```
C:\Users\Jason\Desktop\Examples\ch...
請問您想比 1>大 還是比 2 ) 小 ?1
發牌中...
您拿到的牌是 磚塊 11
電腦拿的牌是 梅花 11
您贏了!
再玩一次嗎? 1>大 2 ) 小 0 ) 不玩了2
發牌中...
您拿到的牌是 黑桃 8
電腦拿的牌是 黑桃 5
您輸了!
再玩一次嗎? 1>大 2 ) 小 0 ) 不玩了0
請按任意鍵繼續 . . .
```

## 11-2 將結構將成參數傳給被呼叫函式(1/2)

---

- ▶ 結構也可做為函式的引數傳入函式之中，延續11-1節宣告的結構，參考下方的函式範例：

```
int cmp(struct animal s1,struct animal s2)
{
    return s1.sex ==s2.sex;
}
```

- ▶ 上方這個範例中，cmp函式接受兩個結構函式[P.11-6上方，應改為結構參數]，如果性別相同就回傳1，反之則回傳0。

## 11-2 將結構變成參數傳給被呼叫函式(2/2)

```
1 /* PassingStructAsArg.c */
2 #include <stdio.h>
3 #include <stdlib.h>
4 struct animal{
5     char name[10];
6     int sex;
7 };
8 int cmp(struct animal , struct animal);
9
10 int main(void)
11 {
12     struct animal dog, cat;
13     printf("請輸入狗的性別：(0:母，1:公)");
14     fflush(stdin);
15     scanf("%d", &dog.sex);
16     printf("請輸入貓的性別：(0:母，1:公)");
17     fflush(stdin);
18     scanf("%d", &cat.sex);
19     printf("Return Value: %d\n", cmp(dog,cat));
20
21     system("pause");
22 }
23 int cmp(struct animal d, struct animal c)
24 {
25     return d.sex==c.sex; //傳回0(等式不成立時)或1(等式成立時)
26 }
```

Line 23 :

被呼叫函式的參數定義(這是傳值的參數傳遞方式)

Line 19 :

呼叫時的參數傳遞

## 11-2 結構做為函式呼叫的傳回值

```
10 int main(void)
11 {
12     struct animal dog, cat;
13     printf("\n讀取狗狗的性別及名字\n");
14     dog=readAnimalData();
15     printf("\n讀取貓咪的性別及名字\n");
16     cat=readAnimalData();
17     printf("\n已輸入的動物清單：\n");
18     printf("-----：\n");
19     printf("性別(0:母，1:公)\t名字\n");
20     printAnimalData(dog, cat);
21     system("pause");
22 }
23 struct animal readAnimalData(){
24     struct animal d;
25     printf("請輸入性別：(0:母，1:公)：");
26     fflush(stdin);
27     scanf("%d", &d.sex);
28     printf("請輸入名字：");
29     fflush(stdin);
30     gets(d.name);
31     return d;
32 }
33 void printAnimalData(struct animal d, struct animal c)
34 {
35     printf("\t%d\t\t%s\n", d.sex, d.name);
36     printf("\t%d\t\t%s\n", c.sex, c.name);
37 }
```

Line 23：

以結構做為傳回值型態

Line 31：

傳回一個結構

Line 14, 16：

將傳回值指定給相關的  
結構變數

註1：

Line 33為傳值的參數傳遞方式

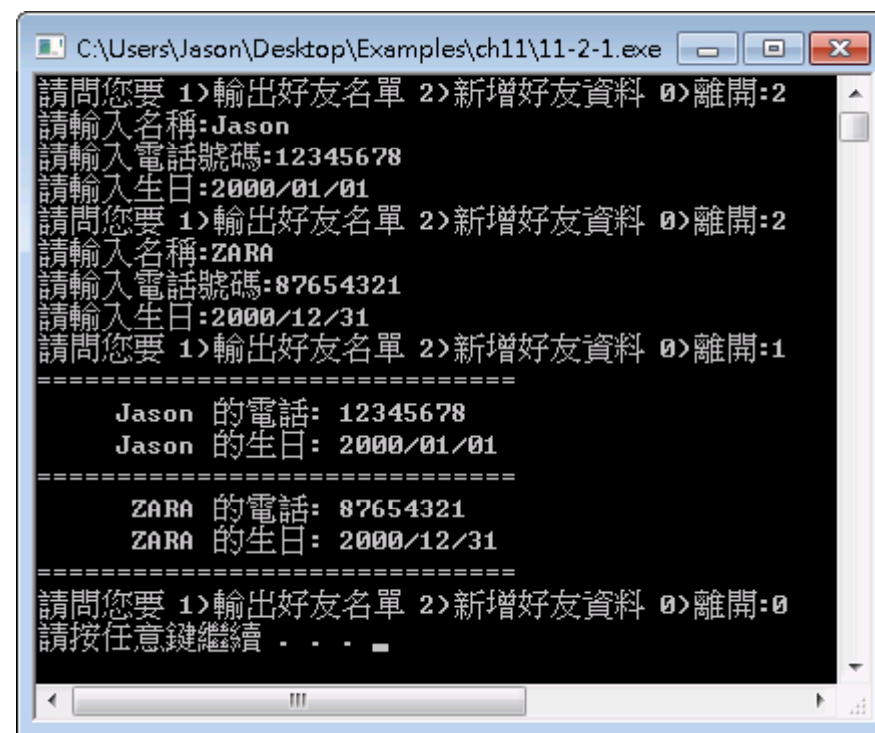
註2：

可將Line 4~Line 9獨自  
放在一個標頭檔中，  
再用#include指令置入  
該檔案



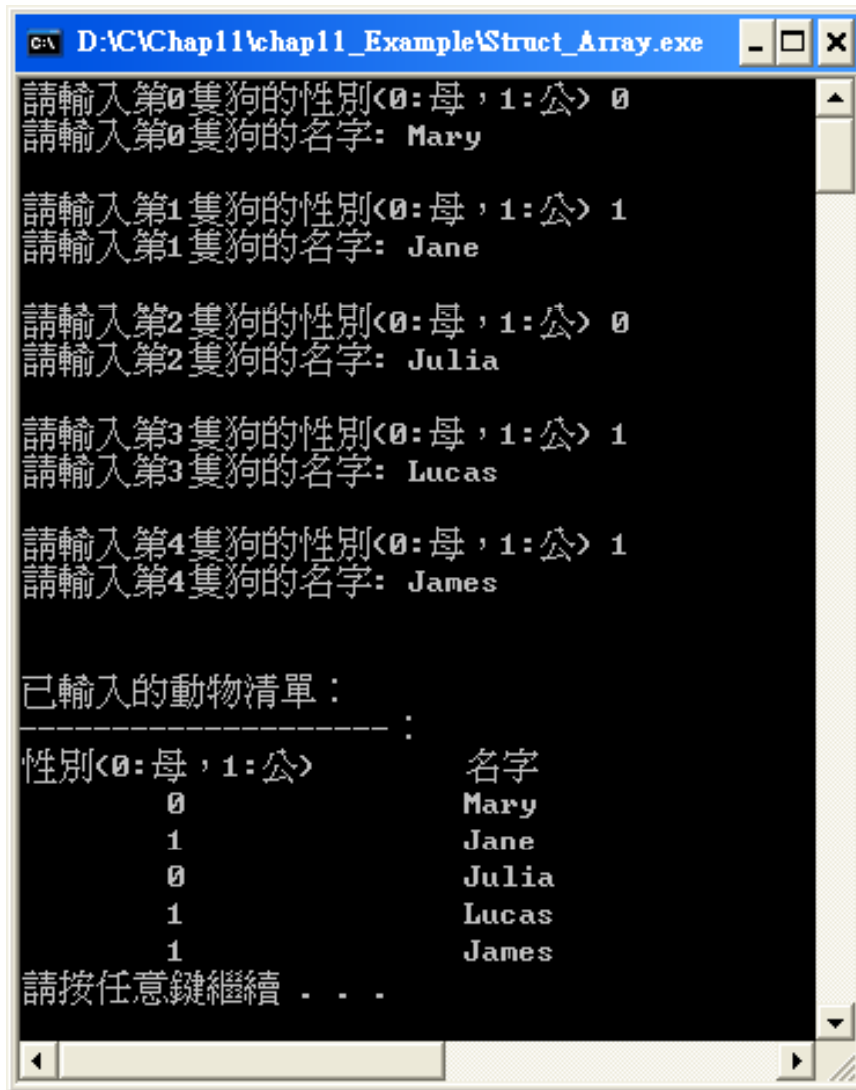
## 程式範例：使用結構設計好友名單程式

- ▶ 學習重點：結構的使用
- ▶ 【參考檔案】11-2-1.c
- ▶ 程式設計目標：撰寫一個程式，定義一個friend結構，此結構儲存好友的資料，內容包括了姓名、電話、生日等三項資料，並撰寫兩個自訂函式，enter()函式可新增一筆好友的資料，show()函式可將所有的好友名單輸出。程式的操作畫面如圖所示。



```
C:\Users\Jason\Desktop\Examples\ch11\11-2-1.exe
請問您要 1>輸出好友名單 2>新增好友資料 0>離開:2
請輸入名稱:Jason
請輸入電話號碼:12345678
請輸入生日:2000/01/01
請問您要 1>輸出好友名單 2>新增好友資料 0>離開:2
請輸入名稱:ZARA
請輸入電話號碼:87654321
請輸入生日:2000/12/31
請問您要 1>輸出好友名單 2>新增好友資料 0>離開:1
=====
Jason 的電話: 12345678
Jason 的生日: 2000/01/01
=====
ZARA 的電話: 87654321
ZARA 的生日: 2000/12/31
=====
請問您要 1>輸出好友名單 2>新增好友資料 0>離開:0
請按任意鍵繼續 . . .
```

## 11-2 結構參數的傳遞：程式練習題



```
C:\D:\C\Chap11\chap11_Example\Struct_Array.exe
請輸入第0隻狗的性別<0:母,1:公> 0
請輸入第0隻狗的名字: Mary
請輸入第1隻狗的性別<0:母,1:公> 1
請輸入第1隻狗的名字: Jane
請輸入第2隻狗的性別<0:母,1:公> 0
請輸入第2隻狗的名字: Julia
請輸入第3隻狗的性別<0:母,1:公> 1
請輸入第3隻狗的名字: Lucas
請輸入第4隻狗的性別<0:母,1:公> 1
請輸入第4隻狗的名字: James

已輸入的動物清單:
-----:
性別<0:母,1:公>      名字
      0              Mary
      1              Jane
      0              Julia
      1              Lucas
      1              James
請按任意鍵繼續 . . .
```

- ▶ 開啟PassingStructArray.c，完成以下程式功能：
  1. 在main函式定義長度為5的結構陣列
  2. 定義readDogData函式，該函式用於**讀取**五隻狗的資料
  3. 定義printDogData函式，該函式用於**輸出**五隻狗的資料
  4. 在main函式中呼叫2, 3所定義的函式
- ▶ 程式執行結果如左列視窗
- ▶ 可用slide 12的實作結果來改寫

## 11-3 結構與指標：定義指向結構的指標

- ▶ 指標可用於指向各種型態的變數，包括結構

- ▶ 宣告語法：

- ▶ 語法一

```
struct 結構名稱 {  
    結構內容;  
} *變數名稱;
```

- ▶ 語法二

```
struct 結構名稱 {  
    結構內容;  
};  
struct 結構名稱 *變數名稱;
```

- ▶ 語法一範例：

```
struct friend {  
    char birth[12];  
    int sex;  
    char name[10];  
} *f1;
```

- ▶ 語法二範例：

```
struct friend {  
    char birth[12];  
    int sex;  
    char name[10];  
};  
struct friend *f1;
```

## 11-3 結構與指標：透過指標存取結構成員

- ▶ 若結構相關定義如下：

```
struct friend {  
    char birth[12];  
    int sex;  
    char name[10];  
};  
struct friend *f1p, f1;  
f1p = &f1; //讓f1p指向f1這個結構
```

- ▶ 存取結構成員的語法：

- ▶ 透過結構變數：

- ▶ 變數名.成員名稱;  
f1.birth;  
f1.sex;  
f1.name;

- ▶ 透過指向結構的指標

- ▶ 變數名->成員名稱;  
f1p->birth;  
f1p->sex;  
f1p->name;

## 11-3 結構與指標:練習題

---

- ▶ 若結構及相關變數定義如下：

```
struct vector {  
    double x;  
    double y;  
    double z;  
};
```

```
struct vector v1, *v1p;
```

```
v1.x = 1.0;
```

```
v1.y = 1.0;
```

```
v1.z = 1.0;
```

- ▶ Q6: 若要讓v1p指向v1，則程式指令應如何寫？
- ▶ Q7: 若要透過v1存取v1的各個結構成員，則語法為？
- ▶ Q8: 若要透過v1p存取v1的各個結構成員，則語法為？

## 11-3 結構與指標:程式練習題

---

- ▶ 開啟readVector.c
- ▶ 依序完成以下程式功能：
  - ▶ 從鍵盤讀取兩個向量
  - ▶ 輸出兩個向量
  - ▶ 輸出兩個向量相加後的向量
  - ▶ 輸出兩個向量相減後的向量

## 被呼叫函式以return指令將結構傳回給呼叫函式

```
1 /* StructAsRetVal_1.cpp */
2 #include <stdio.h>
3 #include <stdlib.h>
4 struct Vector {
5     double x;
6     double y;
7     double z;
8 };
9 struct Vector VectorAddition(struct Vector, struct Vector);
10 int main(void)
11 {
12     struct Vector v1={1,2,3};
13     struct Vector v2={4,5,6};
14     struct Vector sum;
15     sum=VectorAddition(v1,v2);
16     printf("(%.11f, %.11f, %.11f)+(%%.11f, %.11f, %.11f)=(%.11f, %.11f, %.11f)\n",
17           v1.x,v1.y,v1.z,v2.x,v2.y,v2.z,sum.x,sum.y,sum.z);
18     system("pause");
19 }
20 struct Vector VectorAddition(struct Vector v1, struct Vector v2){
21     struct Vector ret_Val;
22     ret_Val.x=v1.x+v2.x;
23     ret_Val.y=v1.y+v2.y;
24     ret_Val.z=v1.y+v2.z;
25     return ret_Val;
26 }
```

## 練習題

---

- ▶ 開啟StructAsRetVal\_1.cpp，
  - ▶ 定義一個函式：
    - ▶ 名稱：vectorSubtract
    - ▶ 參數列：兩個struct Vector型態的參數
    - ▶ 傳回值型態： struct Vector
  - ▶ 在main函式呼叫vectorSubtract 並輸出相量相減結果



## 以傳址的參數傳遞方式傳回運算結果

```
1  /* StructAsRetVal_2.cpp */
2  #include <stdio.h>
3  #include <stdlib.h>
4  struct Vector {
5      double x;
6      double y;
7      double z;
8  };
9  void VectorAddition(struct Vector, struct Vector, struct Vector *);
10 int main(void)
11 {
12     struct Vector v1={1,2,3};
13     struct Vector v2={4,5,6};
14     struct Vector sum;
15     VectorAddition(v1,v2, &sum);
16     printf("(%.11f, %.11f, %.11f)+(%%.11f, %.11f, %.11f)=(%.11f, %.11f, %.11f)\n",
17           v1.x,v1.y,v1.z,v2.x,v2.y,v2.z,sum.x,sum.y,sum.z);
18     system("pause");
19 }
20 void VectorAddition(struct Vector v1, struct Vector v2, struct Vector *v3){
21
22     v3->x=v1.x+v2.x;
23     v3->y=v1.y+v2.y;
24     v3->z=v1.y+v2.z;
25 }
```

## 練習題

---

- ▶ 開啟StructAsRetVal\_2.cpp，
  - ▶ 定義一個函式：
    - ▶ 名稱：vectorSubtract
    - ▶ 參數列：兩個struct Vector型態的參數、一個指向struct Vector的指標
    - ▶ 傳回值型態： void
  - ▶ 在main函式呼叫vectorSubtract 並輸出相量相減結果

## 以傳遞陣列的方式傳回多個執行結果

```
1  /* StructAsRetVal_3.cpp */
2  #include <stdio.h>
3  #include <stdlib.h>
4  struct Vector {
5      double x;
6      double y;
7      double z;
8  };
9  void VectorComputation(struct Vector[]);
10 int main(void)
11 {
12     struct Vector v[4]={{1,2,3},{4,5,6},
13                         {0,0,0},{0,0,0}};
14     VectorComputation(v);
15     printf("(%.11f, %.11f, %.11f)+(%.11f, %.11f, %.11f)=(%.11f, %.11f, %.11f)\n",
16           v[0].x,v[0].y,v[0].z,v[1].x,v[1].y,v[1].z,v[2].x,v[2].y,v[2].z);
17     system("pause");
18 }
19 void VectorComputation(struct Vector v[]){
20
21     v[2].x=v[0].x+v[1].x;
22     v[2].y=v[0].y+v[1].y;
23     v[2].z=v[0].z+v[1].z;
24 }
```

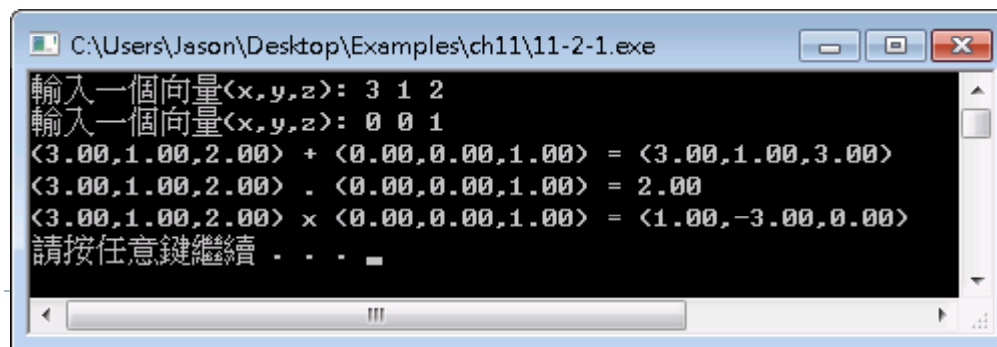
## 練習題

---

- ▶ 開啟StructAsRetVal\_3.cpp，修改VectorComputation函式：
- ▶ 計算v[0]與v[1]兩個向量相減的結果，並將結果指定給v[3]
- ▶ 在main函式中輸出相減結果

## 程式範例：向量運算器程式

- ▶ 學習重點：使用指標傳遞結構入函式
- ▶ 【參考檔案】11-3-1.c
- ▶ 程式設計目標：撰寫一個程式，定義一個三維空間的向量結構vector，使用者輸入兩個向量，程式依次輸出相加、內積與外積三種運算的結果，程式中使用指標傳遞結構入函式，計算後再回傳結果。執行結果如下圖所示。



```
C:\Users\Jason\Desktop\Examples\ch11\11-2-1.exe
輸入一個向量(x,y,z): 3 1 2
輸入一個向量(x,y,z): 0 0 1
<3.00,1.00,2.00> + <0.00,0.00,1.00> = <3.00,1.00,3.00>
<3.00,1.00,2.00> . <0.00,0.00,1.00> = 2.00
<3.00,1.00,2.00> x <0.00,0.00,1.00> = <1.00,-3.00,0.00>
請按任意鍵繼續 . . .
```

## 11-4 聯合(略)

---

- ▶ 聯合型態與結構的宣告方式相似，其宣告的語法如下：

```
union 聯合名稱 {  
    聯合成員1;  
    聯合成員2;  
    ...  
} 聯合名稱;
```

- ▶ 聯合的所有成員會使用同一塊記憶體空間，成員間不需要是相同型態，但同一時間只能以一種型態來解釋該塊記憶體。

## 11-5 列舉型態：宣告方式

---

- ▶ 但變數的值侷限在有限集合內，且程式希望以常數名稱來代表這些可能值時，就可以將該變數型態定義成列舉型態，例：
  - ▶ 用於儲存血型的變數
  - ▶ 用於儲存月份的變數
- ▶ enum 列舉名稱 {整數數1,整數數2,...} 變數名稱;
- ▶ 例：  
`enum BLOODTYPE {A, B, O, AB} bt1;`
- ▶ 程式範例：enumVar.c

## 11-5 列舉型態：程式範例

```
1 /* enumVar.c */
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 enum BLOODTYPE {A, B, O, AB} bt1;
6
7 int main(void)
8 {
9     printf("請輸入血型(A:0, B:1, O:2, AB:3) : ");
10    scanf("%d", &bt1);
11    switch (bt1){
12        case A:
13            printf("A 型\n");
14            break;
15        case B:
16            printf("B 型\n");
17            break;
18        case O:
19            printf("O 型\n");
20            break;
21        case AB:
22            printf("AB 型\n");
23            break;
24    }
25    system("pause");
26    return 0;
27 }
```

### ► Line 5

- 定義列舉型態（可視為一種使用者自訂的資料型態）
- 宣告變數

### ► Line 12, 15, 18, 21： 用名稱來代表變數的內容值，讓程式可讀性更佳



## 11-6 typedef指令

---

- ▶ 用於為既有的資料型態賦予一個新名稱，使用者可以用新的名稱去宣告變數。

- ▶ 語法

typedef 舊型態名稱 新型態名稱;

- ▶ 範例：

```
typedef unsigned int UI;  
UI a;
```

- ▶ 上述範例賦予unsigned int 一個新名稱: UI, 接著宣告一個型態為UI, 名稱為a的變數

## 11-6 typedef指令：程式練習

---

- ▶ 開啟readVector.c，以 typedef 指令賦予下列結構另一個名稱- V，接著以V為型態名稱來宣告相關的變數