

C 程式語言 5

二維陣列

- 二維陣列的宣告

二維陣列的宣告格式

資料型態 陣列名稱 [列的個數] [行的個數];

- 二維陣列宣告的範例：

```
int data[10][5];    /* 可存放10列5行個整數 */  
float score[4][3]; /* 可存放4列3行個浮點數 */
```

表格與二維陣列

- 二維的表格很適合用陣列來儲存：業績以二維陣列表示

業務員	2004 年銷售量			
	第一季	第二季	第三季	第四季
1	30	35	26	32
2	33	34	30	29

陣列 **sale**

	第 1 行	第 2 行	第 3 行	第 4 行
第 1 列	(0, 0) 30	(0, 1) 35	(0, 2) 26	(0, 3) 32
第 2 列	(1, 0) 33	(1, 1) 34	(1, 2) 30	(1, 3) 29

每一格代表一個元素，每個元素皆為 **int** 型態

➡ `int sale[2][4]={{30,35,26,32},
 {33,34,30,29}};`

二維陣列元素的存取

- 利用巢狀迴圈依序輸入二維陣列的元素：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i,j,sale[2][4],sum=0;

    for(i=0;i<2;i++)
        for(j=0;j<4;j++)
        {
            printf("sales%d %d quarter results:",i+1,j+1);
            scanf("%d",&sale[i][j]);          /* 輸入銷售量 */
        }
}
```

二維陣列元素的存取

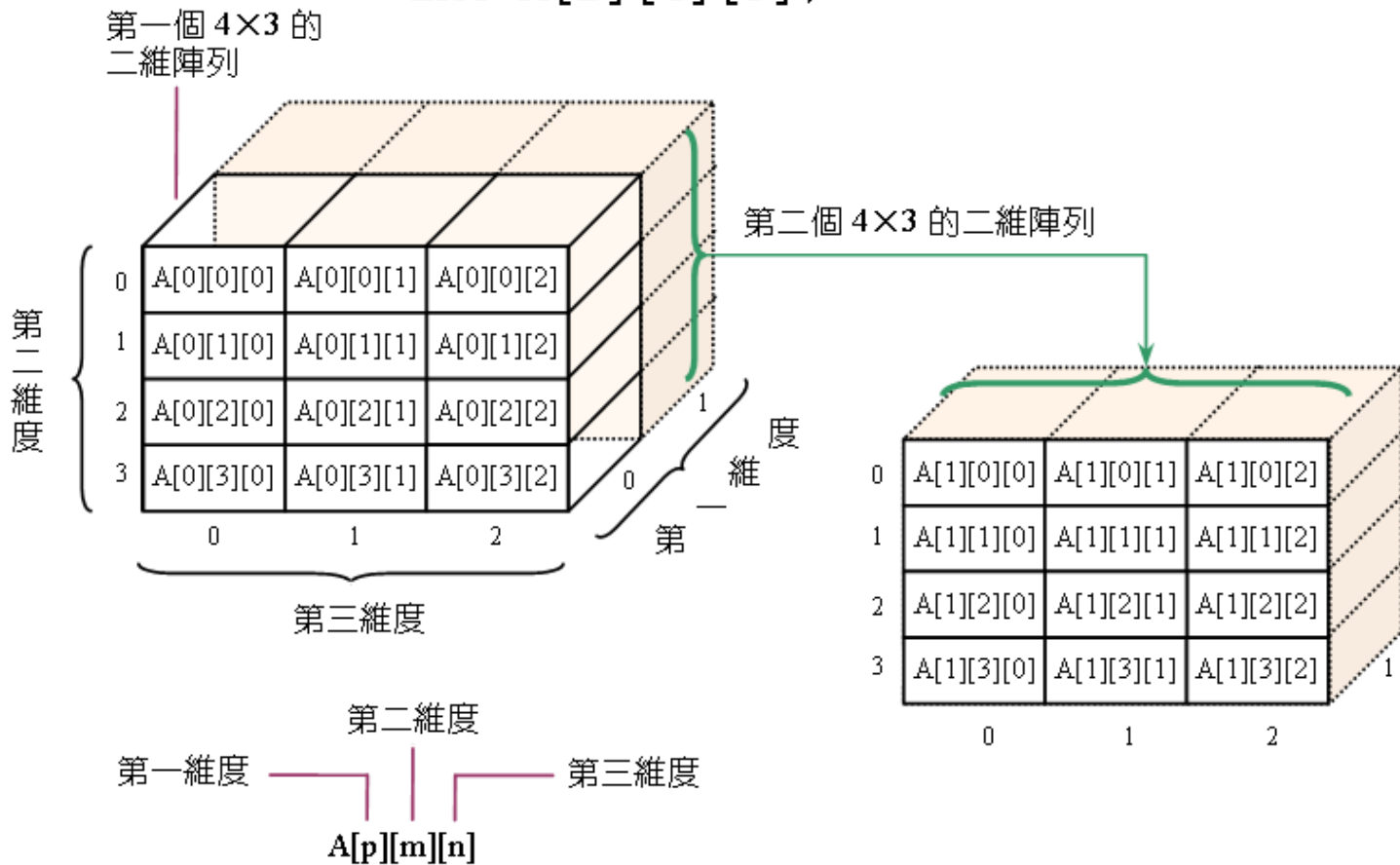
```
printf("***Output***");
for(i=0;i<2;i++)                                /* 輸出銷售量並計算總銷售量 */
{
    printf("\nsales%d quarter results:",i+1);
    for(j=0;j<4;j++)
    {
        printf("%d ",sale[i][j]);
        sum+=sale[i][j];
    }
}
printf("\n2015 results for the total sales volume of %d cars \n",sum);

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

多維陣列

- 三維陣列的結構（以 $2 \times 4 \times 3$ 的陣列為例）：

```
int A[2][4][3];
```



三維陣列

- 找出三維陣列裡，所有元素的最大值

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
```

```
    int A[2][4][3]={{ {21, 32, 65},
                        {78, 94, 76},
                        {79, 44, 65},
                        {89, 54, 73}},
                     {{32, 56, 89},
                        {43, 23, 32},
                        {32, 56, 78},
                        {94, 78, 45}}};
```

第一個 4×3 的
二維陣列

第二個 4×3 的
二維陣列

第二個 4×3 的二維陣列

0	32	56	89
1	43	23	32
2	32	56	78
3	94	78	45

第一個 4×3 的
二維陣列

0	21	32	65
1	78	94	76
2	79	44	65
3	89	54	73

由中層迴圈
來控制

第三維度

由內層迴圈
來控制

由外層迴圈
來控制

字串的宣告與初值的設定

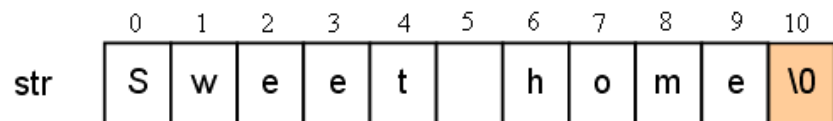
- 字元以單引號包圍，而字串則是以雙引號包圍：
 - 'a' /* 這是字元常數 a */
 - "a" /* 這是字串常數 a */
 - "Sweet home" /* 這是字串常數 Sweet home */

- 下面是字串宣告的語法：

字串宣告的語法

char 字元陣列名稱[陣列大小] = 字串常數;

char str[]="Sweet home";



字串結束符號

字元與字串之比較

- 字元與字串之比較的範例：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char ch='a';                /* 宣告字元變數ch */
    char str1[]="a";            /* 宣告字串變數str1 */
    char str2[]="Sweet home";   /* 宣告字串變數str2 */

    printf("ch:%d byte\n",sizeof(ch));
    printf("str1:%d byte\n",sizeof(str1));
    printf("str2:%d byte\n",sizeof(str2));

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

字串的輸入與輸出函數

- `gets()` 與 `puts()` 的格式：

`gets()` 的格式

`gets (字元陣列名稱) ;`

`puts()` 的格式

`puts (字元陣列名稱) ;`

或者是

`puts (字串常數) ;`

字串陣列

- 字串陣列的宣告與初值設定的格式：

字串陣列的宣告

```
char 字元陣列名稱[字串的個數][字串長度];
```

宣告字串陣列，並設定初值

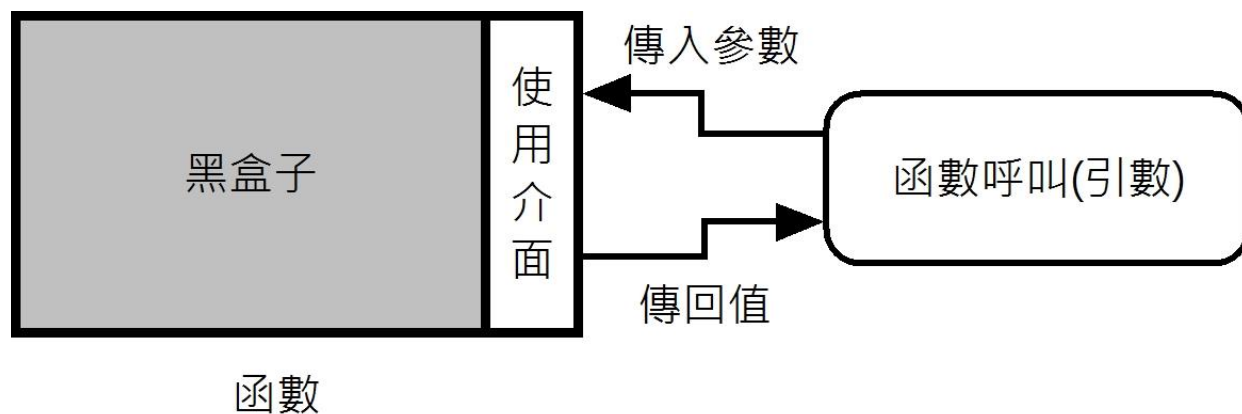
```
char 字元陣列名稱[字串的個數][字串長度]=  
    {"字串常數1", "字串常數2", ..., "字串常數n"};
```

```
char customer[6][15];  
char S[3][10]={"Tom","Lily","James Lee"};
```

C 語言的函數

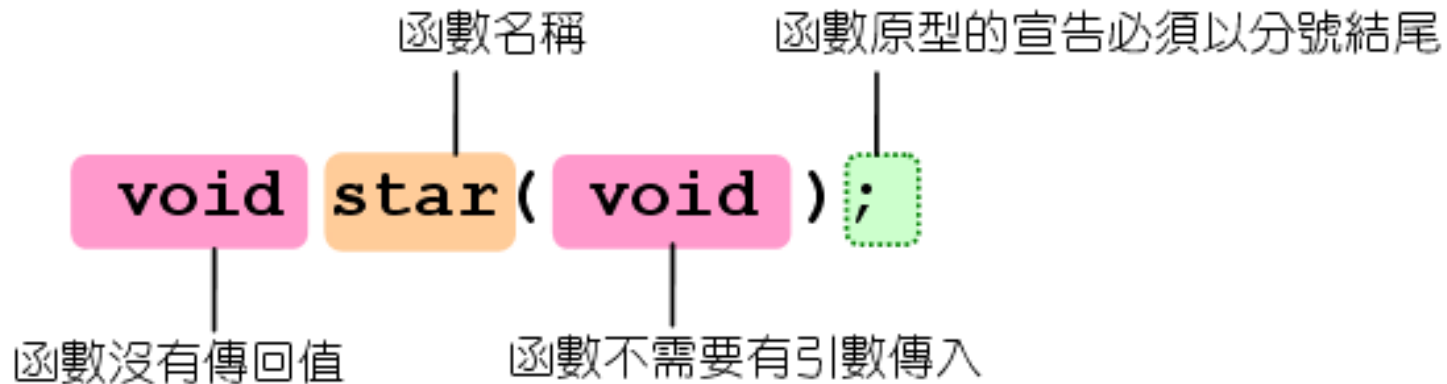
- 目的是要達到單純化，程式模組化
 - 將大問題細分成小問題
 - 將解決這些小問題的方法，撰寫成較小的程式區塊
- C 語言的函數
 - 賦予程式區塊一個名字
 - 指定它的輸出與輸入
 - 此程式區塊就是一個 C 語言的函數

- 函數是一個獨立功能的程式區塊，如同是一個「黑盒子」（**Black Box**），我們根本不需要了解函數定義的程式碼內容，只要告訴我們如何使用此黑盒子的「介面」（**Interface**），就可以呼叫函數來使用函數的功能，如下圖所示：



star() 函數原型的宣告

- star() 函數原型宣告的語法：



建立C語言的函數

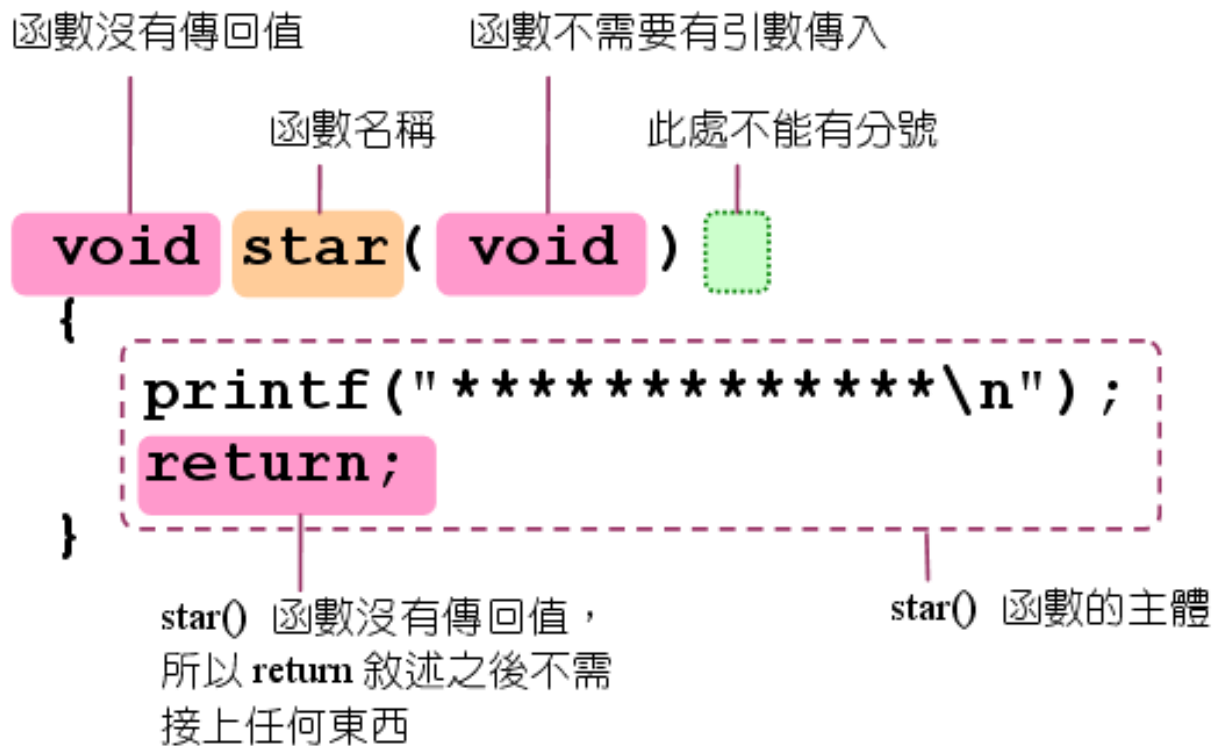
- C語言的使用者自訂函數，它是由函數標頭和程式區塊所組成，其基本語法如下所示：

```
傳回值型態 函數名稱(引數){  
    程式敘述;  
    .....  
    return;  
}
```

- 上述函數的第1列是函數標頭（Function Header），之後的大括號是函數的程式區塊（Function Block）。

star() 函數原型的定義

- star() 函數的定義：



簡單的函數範例

- **star()** 函數可印出一行星號

```
#include <stdio.h>
#include <stdlib.h>
void star(void);                                /* star()函數的原型 */
int main(void) {
    star();                                     /* 呼叫star函數 */
    printf(" Welcome to the C language \n");
    star();                                     /* 呼叫star函數 */
    system("pause");
    return 0;
}

void star(void)
{
    printf("*****\n");                        /* 印出13個星號 */
    return;
}
```

函數的基本架構

- 函數原型宣告

傳回值型態 函數名稱(引數型態1, 引數型態2,...);

- add() 可接收二整數，傳回值為整數之和，其原型為：

傳回值型態是整數

有兩個引數傳入 add() 函數，型態均為 int，各型態間以逗號分開

```
int add ( int , int );
```

函數名稱為 add

The diagram illustrates the components of the function prototype `int add (int , int);`. The return type `int` is highlighted in an orange box, with a vertical line pointing to the annotation '傳回值型態是整數' (Return value type is integer). The function name `add` is highlighted in a pink box, with a vertical line pointing to the annotation '函數名稱為 add' (Function name is add). The parameters `(int , int)` are grouped by a purple bracket, with a line pointing to the annotation '有兩個引數傳入 add() 函數，型態均為 int，各型態間以逗號分開' (Two arguments are passed to the add() function, both of type int, separated by a comma).

函數的基本架構

- 函數的定義

傳回值型態 函數名稱(型態1 引數1, ..., 型態n 引數n)

```
{  
    變數宣告;  
    敘述主體;  
    return 運算式; /* 傳回運算式的值 */  
}
```

add() 函數的定義：

傳回值型態是整數

傳入的引數分別由變數 a 與 b 接收

```
int add ( int a, int b )  
{  
    retrun a+b;  
}
```

a+b 是整數，所以傳回值的型態是整數

於程式裡呼叫函數

- add() 函數的使用範例：

```
#include <stdio.h>
#include <stdlib.h>
int add(int,int);          /* add()函數的原型 */
int main(void) {
    int sum, a=5, b=3;
    sum=add(a,b);          /* 呼叫add()函數，並把傳回值設給sum */
    printf("%d+%d=%d\n",a,b,sum);

    system("pause");
    return 0;
}

int add(int num1, int num2) /* add()函數的定義 */
{
    int a;                  /* 於add()函數裡宣告變數a */
    a=num1+num2;
    return a;               /* 傳回num1+num2 的值 */
}
```

-The End-