Pointer: 指標

回 什麼是指標?

指標(pointer)其實是一種特殊的變數，用來存放變數在記憶體中的位址。

回 為什麼要使用指標?

1．利用指標可以使得函數在傳遞陣列或字串時更有效率。
2．較複雜的資料結構，如鏈結串列（linked list）或二元樹（binary tree）等，均須要指標的協助才能將資料鏈結在一起。
3．許多函數必須利用指標來傳達記憶體的訊息，例如記憶體配置函數 malloc()與檔案開啓函數 fopen()等，都必須借助指標的幫忙。

回 指標變數

(一) 宣告

```
int     *ptr1;                    //宣告一個名爲 ptr1 的整數型態之指標變數
char    *ptr2;                    //宣告一個名爲 ptr2 的字元型態之指標變數
```

(二) 使用

位址運算子 **&** ：用來求取變數的位址。

取值運算子 ***** ： 用來取得指標所指向的記憶體位址的內容。

例如：

```
int    *ptri;
int    a=10;
ptri = &a;
printf("ptri 所指的位址上的內容爲%d",*ptri);
```

(三)宣告與使用一定要相匹配喔!

指標變數的型態要和所指向的變數型態相同。

回 【練習一】:

1、假設 ptr 是指向整數變數 i 的指標，下列何者正確? Ans: _____

(a) i = *ptr;
(b) ptr = &i;
(c) ptr = *i;
(d) *ptr = i;

2、於下表中，假設 a 的位址爲 1000，b 的位址爲 2000，請依序將下列空格填滿，並寫一程式，印出表格裏相關的數據，用以驗證您所填入的數值。

	程式碼	a	b	ptr	*ptr
1	int a = 12, b=7;	12	7		
2	int *ptr;				
3	ptr = &a;				
4	*ptr = 19;				
5	ptr = &b;				
6	b = 16;				
7	*ptr = 12;				
8	a = 17;				
9	ptr = &a;				
10	a = b;				
11	*ptr = 63;				

3、於下表中，假設 a 的位址爲 1000，b 的位址爲 2000，請依序將下列的空格填滿，並撰寫一程式，印出表格裏的相關數據，用以驗證您所填入的數值：

	程式碼	a	b	ptr1	*ptr1	ptr2	*ptr2
1	int a = 28, b=16;	28	16				
2	int *ptr, *ptr2;						
3	ptr1 = &b;						
4	ptr2 = &a;						
5	*ptr1 = 4;						
6	a = 16;						
7	*ptr2 = 12;						
8	ptr2 = ptr1;						
9	*ptr1 = 19;						
10	Ptr1 = &a;						
11	a = 7;						
12	*ptr2 = *ptr1						

● 指標與函數

在函數中，若是想傳回某個結果給原呼叫函數，可以用 return 敘述，但是，return 敘述只能傳回一個值!!

所以，當程式中需要傳遞兩個以上的值，就無法利用 return 敘述。

此時，指標就可以解決函數間傳遞多個傳回值的問題。

【練習二】：請寫下執行結果(並說明)

#include <stdio.h>
void swap(int,int);
int main(void)
{
int a=3,b=5;
 	
printf("Before swap...");
printf("a=%d,b=%d\n",a,b);
printf("After swap...");
swap(a,b);
printf("a=%d,b=%d\n",a,b);
return 0;
}
void swap(int x , int y)
/* 將兩數互換 */
{
int temp=x;
 	
x=y;
y=temp;
return;
}
	#include <stdio.h>
	void swap(int *,int *);
	int main(void)
	{
	int a=3,b=5;
	
	printf("Before swap...");
	printf("a=%d,b=%d\n",a,b);
	printf("After swap...");
	swap(&a,&b);
	printf("a=%d,b=%d\n",a,b);
	return 0;
	}
	void swap(int *x , int *y)
	/* 將兩數互換 */
	{
	int temp=*x;
	
	*x=*y;
	*y=temp;
	return;
	}

CALL by VALUE (傳值呼叫)	**Call by address(傳址呼叫)**
執行結果：	執行結果：
說明：	說明：

回 練習三：請 Trace 結果：

#include <stdio.h>
void rect(int,int,int *,int *);
int main(void)
{
int a=5,b=8;
int area,length;
 	
rect(a,b,&area,&length);
/* 計算面積及邊長 */
printf("area=%d,total length=%d\n",area,length);
return 0;
}
	void rect(int x , int y, int *ptr1, int *ptr2)
	{
	*ptr1=x*y;
	ptr2=2(x+y);
	return;
	}

請問： area= _____, length=_____	

指標與陣列

陣列：是固定長度的記憶體區塊。
指標：是實是一種特殊的變數，用來記錄所指向變數的位址。
陣列的元素排列可以利用指標的運算來存取。

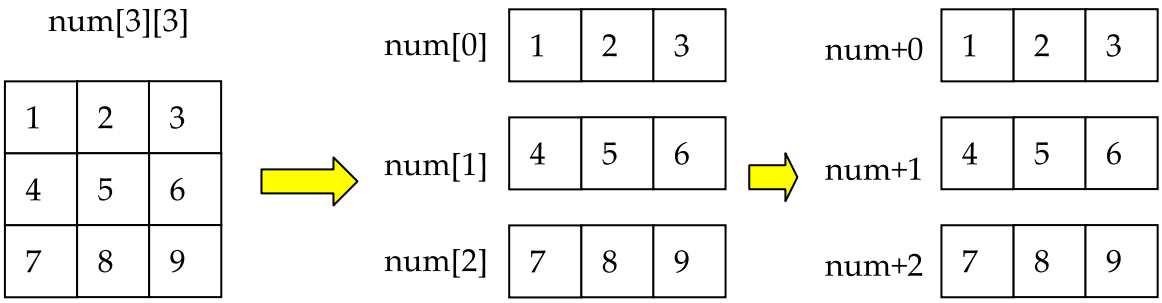
回 指標與陣列的關係

1．陣列是利用註標取出元素值。

2．若宣告 int a[3]={5,7,9};
則指標 a，所指的是 a[0]的位址。 即，
*a 和 a[0]的值是一樣的。
*(a +1) 和 a[1] 的值是一樣的。
*(a + 2) 和 a[2] 的值是一樣的。

回 二維陣列 vs. 雙重指標

若宣告 int num[3][3];



num[0][0] *(*(num+0)+0)
num[0][1] _____
num[0][2] _____
num[1][0] _____
num[1][1] _____
num[1][2] _____
num[2][0] _____
num[2][1] _____
num[2][2] _____

練習：
如果程式裏有如下的宣告：
int arr[2][4] = {{2,3,4,5},{6,7,8,9}};
假設 arr[0][0]的位址爲 1200，試回答下列問題：
(a) arr 的值爲何?
(b) arr[0]與 arr[1]的值各是多少?
(c) arr + 1 的值爲何?
(d) *(arr+0) 與 *(arr+1)的值爲何?
(e) *(arr+1)+ 0 = _____
*(arr+1)+ 1 = _____
(f) *(*(arr+1)+ 0) = _____
((arr+1)+ 1) = _____*(*(arr+1)+ 2) = _____

回 【練習一】:

```
#include <stdio.h>
int main(void)
{
    int a[3]={5,7,9};
    int i,sum=0;

    for(i=0;i<3;i++)
        sum+=*(a+i);
    printf("sum=%d\n",sum);
    return 0;
}
```

請畫下過程：

I = 0 :

I = 1:

I = 2:

最後，sum = _____

回 【練習二】:利用指標，將陣列傳入函數。

請寫下執行結果(並說明)

```
#include<stdio.h>
void add(int *,int,int,int);
int main(void)
{
    int a[10]={1,2,3,4,5,6,7,8,9,10};
    int i,n=3,m=8,k=5;

    printf("Before process...\narray a=");
    for(i=0;i<10;i++)                          /* 印出陣列的內容 */
        printf("%3d",a[i]);

    add(a,n,m,k);                              /* 呼叫自訂函數 add() */
    printf("\nAfter process...\narray a=");
    for(i=0;i<10;i++)                           /* 印出陣列的內容 */
        printf("%3d",a[i]);

    return 0;
}

void add(int *ptr,int n,int m,int k)
{
    int i;

    for(i=n;i<=m;i++)
        *(ptr+i)+=k;
    return;
}
```

執行結果：

回　函數傳回值為指標型態的情況。

練習三：請 Trace 結果：

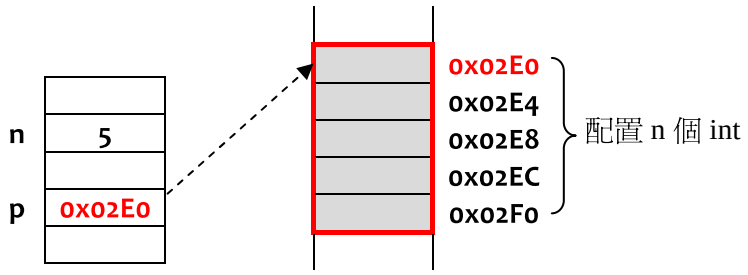
```
#include <stdio.h>
#define SIZE 5
int *maximum(int *);
int main(void)
{
    int a[SIZE]={3,1,7,2,6};
    int i,*ptr;

    printf("array a=");
    for(i=0;i<SIZE;i++)
        printf("%d ",a[i]);
    ptr=maximum(a);
    printf("\nmaximum=%d\n",*ptr);
    return 0;
}
```

```
/*  尋找並傳回陣列的最大值  */
int *maximum(int *m)
{
    int i,*max;

    max=m;
    for(i=1;i<SIZE;i++)
        if(*max < *(m+i))
            max=m+i;
    return max;
}
```

動態配置記憶體：

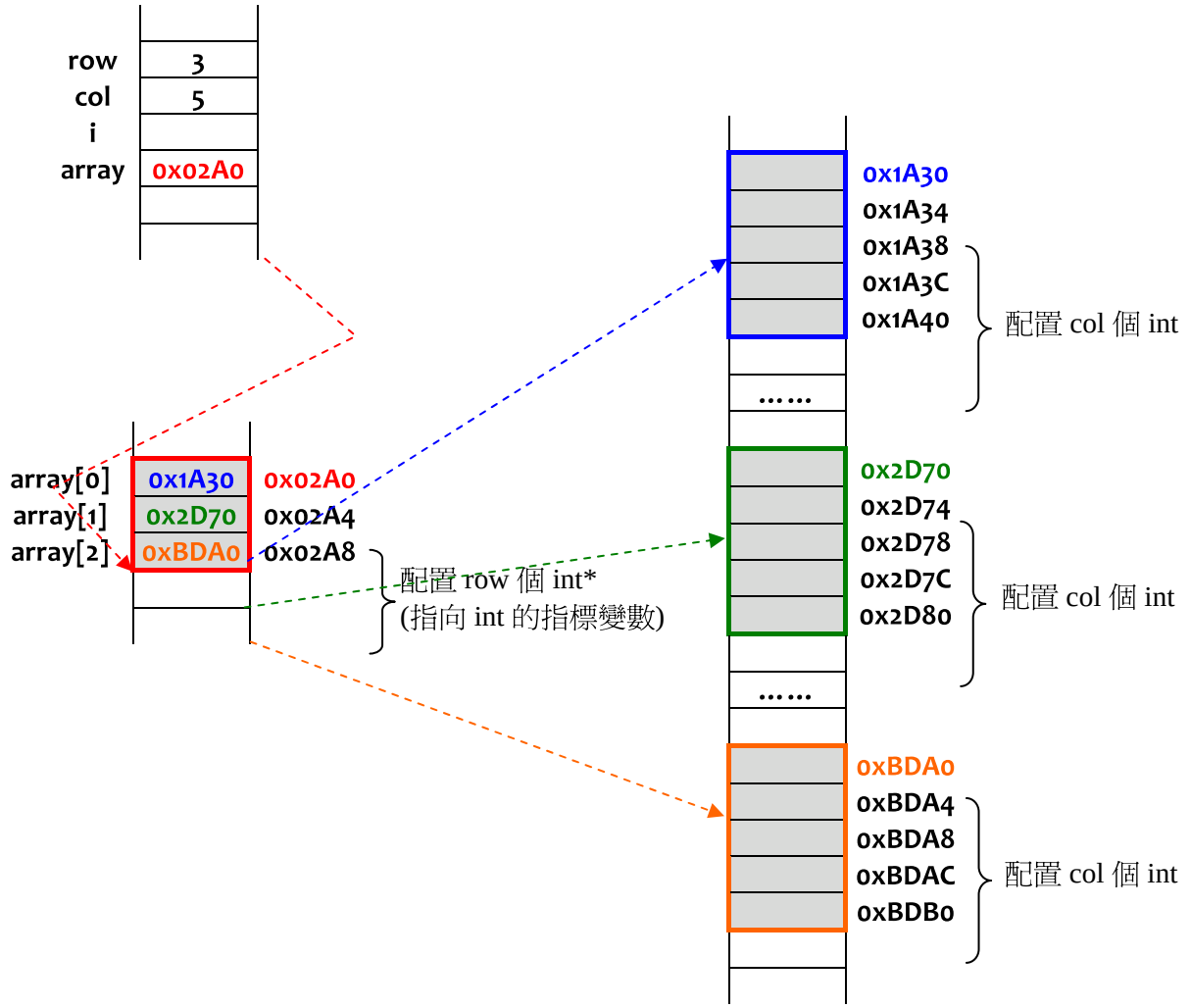
一維配置

```
int n=5;
int* p;

p = (int*) malloc( n * sizeof(int) );
```



二維配置

```
int row=3,col=5,i;
int** array;

array = (int**) malloc(row * sizeof(int*));
for (i=0; i<row; i++)
        array[i] = (int*) malloc(col * sizeof(int));
```



範例一：

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int *score;
    int i;
    int num;
    int sum;
    float ave;

    sum=0;
    printf("請輸入學生人數:");
    scanf("%d",&num);
```

```
    score = (int *) malloc( num * sizeof(int));
    if (!score)
    {
      printf("Memory allocate fail!! \n");
      exit(1);
    }
    for (i=0; i<num; i++)
    {
    printf("Input    English score.    >>>");
    scanf("%d", &score[i]);
    sum += *(score + i);
    }
    ave = (float) sum / (float) num;
    printf("The average is %6.2f \n",ave);

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