

# 程式設計與實習(一)

---

臨時被叫來上課之熬夜趕工版

BY 孫茂勛

EMAIL:JOHN85051232@GMAIL.COM

# 首先

---

關於自己期中考成績...想知道的請於下課來看

有問題的再留下來討論



# 再來

---

自己回去複習一下一些重要觀念：

1 · 一維陣列的宣告：

<http://openhome.cc/Gossip/CppGossip/OneDimArray.html>

2 · 二維陣列是什麼？

3 · Function 的參數是什麼意思？

# 再來

複習一下一些重要觀念：

4 · 找眾數、質數作業等題日常用到的一個技巧：建表  
(把陣列當成一個查詢的表格)

找眾數:陣列的Value儲存出現次數

找質數:陣列的Value儲存是否是質數(Ex:1代表不是質數、0代表是質數)

Index	0	1	2	3	4
Value	0	0	0	0	0

如此一來，最後只要使用一個迴圈判斷陣列  
各位置的值就可以得到答案。



## 警告

---

以下部份很抽象很難懂  
回去請多練習及上網查資料或詢問我

# Pointer

---

學習指標前的幾件事：

- 指標跟記憶體有關係
- 只要打錯整個程式就很容易當掉，所以不要緊張
- C 語言的特色之一

# Pointer

我們先來看一下變數的宣告：

```
int a = 10;
```

這一句對電腦代表甚麼意思呢？

a	
記憶體	0x000001
值	10

讓電腦生出一個4Byte大小的  
記憶體空間  
然後把10這個資料存進去

# Pointer

再來看一下變數的指定：

`a = 20;`

這一句對電腦代表甚麼意思呢？

a	
記憶體	0x000001
值	20

讓電腦把0x000001這個位置的資料改成20



# Pointer

```
#include <stdio.h>
#include <stdlib.h>
/*pointer*/
int main()
{
    int a = 10;
    float b = 1.5;
    printf( "%p\n", &a);
    printf( "%p\n", &b);
    return 0;
}
```

0043FD08

0043FCFC


請按任意鍵繼續 . . .

# Pointer

今天有一種變數可以儲存某個記憶體位置，叫做指標變數

Ex : `int *b = &a;` //宣告**b**是個指標變數，指向(儲存)**a**變數的記憶體位置

記憶體	0x000001	0x000005
值	10	0x000001



變數**b**有自己的記憶體位址，但儲存的值是**a**的記憶體位址。  
知道**a**變數的記憶體位址，我們就可以透過**b**去修改**a**。

# Pointer

---

Pointer (指標)中兩個重要的運算子：

- 取值運算子\*：用來取得指標變數的值。
- 取址運算子&：取出某個變數的記憶位址。
- Ex：scanf( "%d" ,&a); 是將資料寫入變數a的記憶體位址，所以前面要加&

# Pointer

指標的資料型態：與變數相同，有int/float/char....

指標的宣告：

```
int i = 10;
```

```
int *ptr = &i;
```

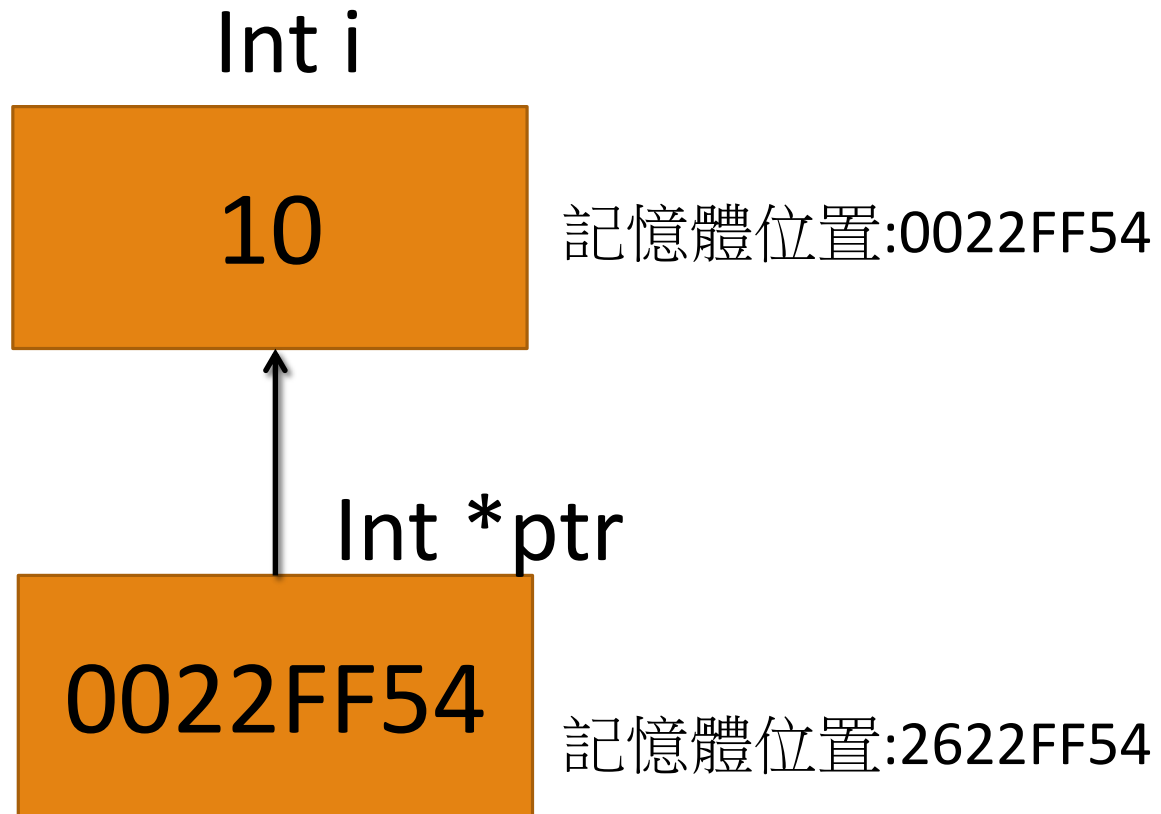
或

```
int i = 10;
```

```
int *ptr = null;
```

...

```
ptr = &i;
```



# Pointer

以下的程式碼執行結果？

```
int a = 10;  
int *ptr = &a; //宣告ptr是一個指標變數，指向a的記憶體位址|  
printf( "%p\n", &*ptr);  
printf( "%p\n", *&ptr);
```

取值 ( \* ) 與取址 ( & ) 是互補的(可以互相抵銷)

# Pointer

```
int num = 10;  
int *ptr = NULL; //宣告一個指標，不指向任何東西  
  
ptr = &num; //ptr指向num的記憶體位址  
  
printf("num的值 : %d\n", num);  
printf("num的記憶體位址 : %p\n", &num);  
  
printf("ptr指向位址的值 : %d\n", *ptr);  
printf("ptr指向位址的記憶體位址 : %p\n", ptr);  
printf("ptr的記憶體位址 : %p\n", &ptr);
```

# Pointer

---

常見問題:

```
int *ptr;  
*ptr = 10; //error
```

沒有給 \*ptr 一個初始的位置，直接給值的話，通常會出現記憶體區段錯誤。

如果宣告完指標沒有要馬上指定變數的話，可以先設初始值為NULL，表示沒有指向任何變數。

# Pointer

---

常見問題:

```
int* prt1, ptr2;
```

這樣的宣告只有ptr1才是指標，ptr2是一個整數變數

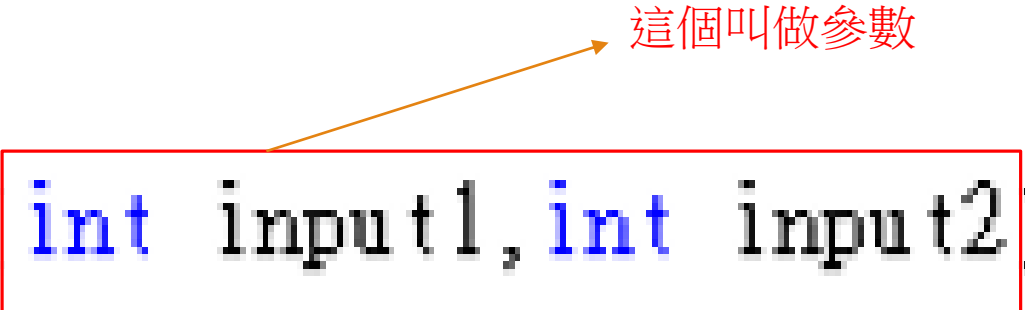


# Pointer

大概知道指標是什麼東西後.....

我們回頭看下之前學過的Function是如何傳遞參數的.....

`void fun(int input1, int input2)`



這個叫做參數

# Pointer

```
void swap(int a , int b)
{
    int temp = 0;
    temp = a;
    a = b;
    b = temp;
}
```

執行結果應該是什麼？

```
int main()
{
    int a = 10;
    int b = 20;
    printf("交換前: a = %d b = %d\n", a, b);
    swap(a, b);
    printf("交換後: a = %d b = %d\n", a, b);
    system("pause");
    return 0;
}
```

# Function

---

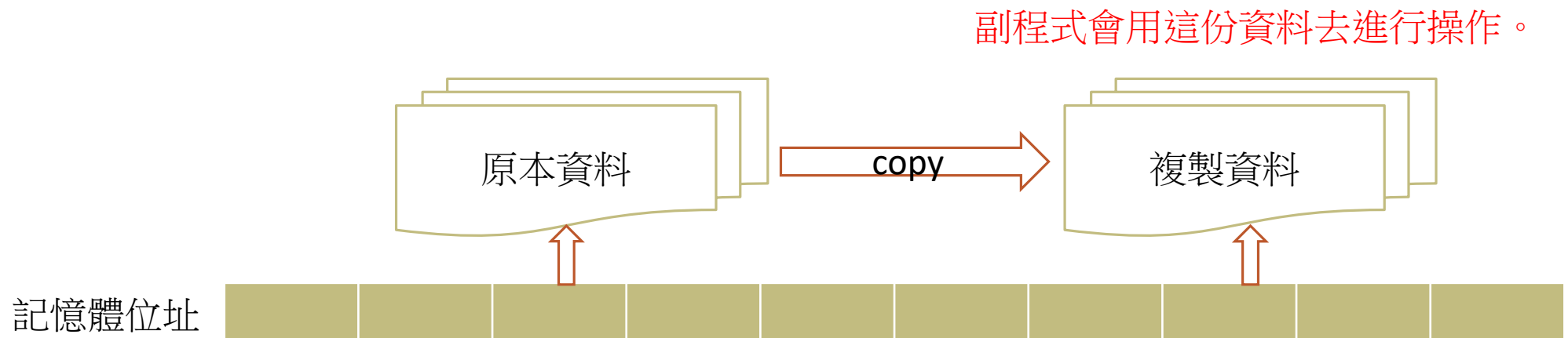
參數的傳遞方式 ( Passing Arguments ) :

- Pass By Value
- Pass By Address
- ~~Pass By Reference(study in C++)~~

# Function

## Pass By Value(傳值)：

- 副程式的參數會將要使用的資料複製一份，副程式會以被複製的資料進行操作。
- 不會影響到原來的資料。



# Function

```
#include <stdio.h>
#include <stdlib.h>
//pass by value
void fun(int input1,int input2)
{
    input1 = input1 + 10;
    input2 = input2 + 10;

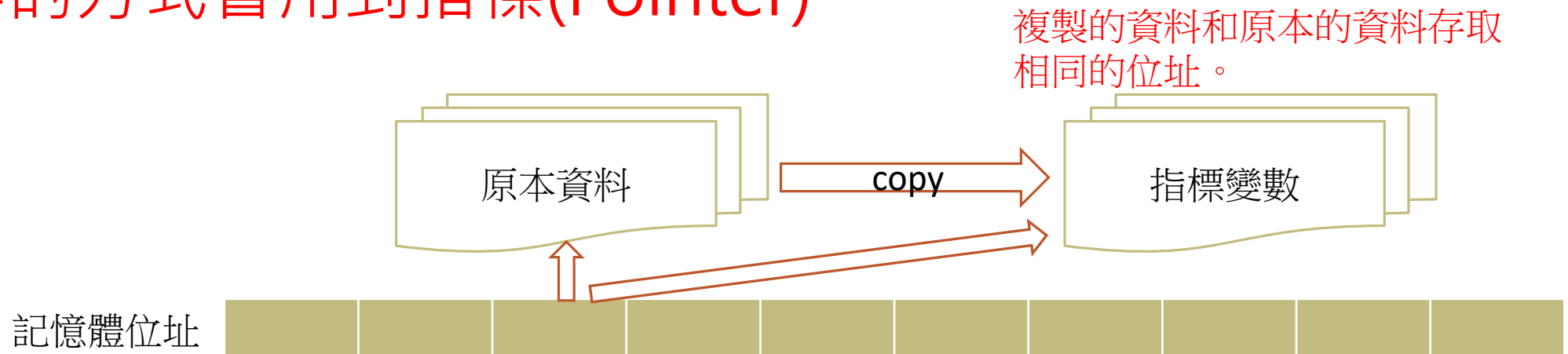
    printf("在function中a的值為:%d\n",input1);
    printf("在function中b的值為:%d\n",input2);
}
int main( )
{
    int a = 5;
    int b = 10;
    printf("在呼叫function前a的值為:%d\n",a);
    printf("在呼叫function前b的值為:%d\n",b);
    fun(a,b);
    printf("在呼叫function後a的值為:%d\n",a);
    printf("在呼叫function後b的值為:%d\n",b);
    system("PAUSE");
    return 0;
}
```

在呼叫function前a的值為:5  
在呼叫function前b的值為:10  
在function中a的值為:15  
在function中b的值為:20  
在呼叫function後a的值為:5  
在呼叫function後b的值為:10  
請按任意鍵繼續 . . .

# Function

## Pass By Address(傳址)：

- 副程式的參數會將要使用資料的記憶體位置複製一份，副程式會使用指標再該位置上進行操作。
- 會影響到原來的資料。
- 操作的方式會用到指標(Pointer)。



# Function

```
#include <stdio.h>
#include <stdlib.h>
//pass by address
void fun(int *input1, int *input2)
{
    *input1 = *input1 + 10;
    *input2 = *input2 + 10;
    |
    printf("在function中a的值為:%d\n", *input1);
    printf("在function中b的值為:%d\n", *input2);
}
int main()
{
    int a = 5;
    int b = 10;
    printf("在呼叫function前a的值為:%d\n", a);
    printf("在呼叫function前b的值為:%d\n", b);
    fun(&a, &b);
    printf("在呼叫function後a的值為:%d\n", a);
    printf("在呼叫function後b的值為:%d\n", b);
    system("PAUSE");
    return 0;
}
```

宣告要傳入的參數是指標變數

呼叫時必須給變數的"記憶體位址"

在呼叫function前a的值為:5  
在呼叫function前b的值為:10  
在function中a的值為:15  
在function中b的值為:20  
在呼叫function後a的值為:15  
在呼叫function後b的值為:20  
請按任意鍵繼續 . . .

# Binary Search

```
int binary_sort(int arr[], int num, int find)
{
    int min = 0;
    int max = num-1;
    int mid = (min + max) / 2;
    while(min <= max)
    {
        if(arr[mid] == find)
        {
            return find;
        }
        else if(arr[mid] < find)
        {
            min = mid + 1;
        }
        else // arr[mid] > find
        {
            max = mid - 1;
        }
        mid = (min + max) / 2;
    }
    return -1;
}
```

告訴function我要傳的參數是一個陣列

```
int main()
{
    int arr[10] = {0,1,2,3,4,5,6,7,8,9};
    printf("%d\n",binary_sort(arr,10,3));
    system("PAUSE");
    return 0;
}
```

一維陣列的參數傳遞需要告知起始位置  
(arr代表arr[0]的記憶體位址)

=>arr和 &arr[0] 是一樣的意思



# Function

---

## ~~Pass By Reference(傳參考)：~~

- 副程式的參數會新增一個連結，連結至原本的資料，副程式會直接對該資料進行操作。
- 會影響到原來的資料。
- 與Pass By address的差別：傳址在實作上會多新增指標去複製記憶體位置，傳參考則是直接創立連結。