


MMS Lab MultiMedia Systems Laboratory

CHAPTER 5

Points(ch7)



NTUT MMS LAB 1

MMS Lab Points

- 指標變數(pointers)主要存放**記憶體位址**的變數，不是存放一般資料內容，主要存放**位址**
- Pointers could make parameter passing of functions more efficient
 - Call by value (傳值)
 - 函數呼叫: function(a, b)
 - 函數定義: void function(int x, int y)
 - Call by address (傳位址)
 - 函數呼叫: function(&a, &b)
 - 函數定義: void function(int *x, int *y)
 - Call by reference (傳參考)
 - 函數呼叫: function(a, b)
 - 函數定義: void function(int &x, int &y)

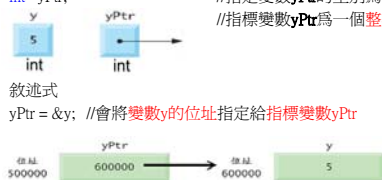
NTUT MMS LAB 2

MMS Lab Points

- *運算子
 - 間接運算子 (indirection operator)或反參考運算子 (dereferencing operator)
 - 傳回其運算元 (即指標) 所指向的物件的數值
- &運算子
 - 取址運算子 (address operator)
 - 傳回運算元的**位址**運算

舉例來說，我們假設以下的定義

```
int y = 5;
int *yPtr; //變數y是一個整數
            //指定變數yPtr的型別為 int *
            //指標變數yPtr為一個整數的指標
```



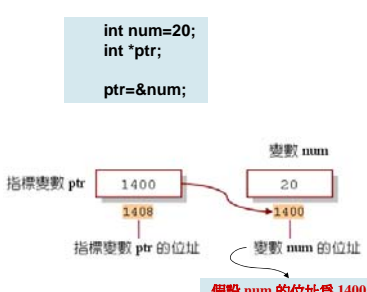
敘述式

yPtr = &y; //會將變數y的位址指定給指標變數yPtr

NTUT MMS LAB 3

MMS Lab Points

- 指標基本運算圖例



NTUT MMS LAB 4

MMS Lab Points

- 整數指標變數(int *)與整數變數(int)

```
#include <stdio.h>
#include <stdlib.h>

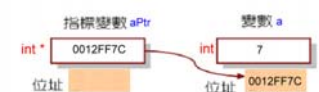
int main(void)
{
    int a;
    int *aPtr;

    a=7;
    aPtr=&a;

    printf("The address of a is %p\n", &a);
    printf("The value of aPtr is %p", &aPtr);

    printf("\n\nThe value of a is %d\n", a);
    printf("\n\nThe value of *aPtr is %d", *aPtr);

    printf("\n\nShowing that * and & are complements of each other\n");
    printf("each other\n\n*aPtr = %p", &a);
    printf("\n\n*aPtr = %p\n", &*aPtr, *aPtr);
    system("pause");
    return 0;
}
```



NTUT MMS LAB 5

MMS Lab Points

- 函數的傳值呼叫 (call by value)

```
#include <stdio.h>
#include <stdlib.h>

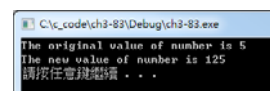
int cubeByValue(int n);

int main(void)
{
    int number=5;
    printf("The original value of number is %d", number);

    number=cubeByValue(number);

    printf("\n\nThe new value of number is %d", number);
    system("pause");
    return 0;
}

int cubeByValue(int n)
{
    return n * n * n;
}
```



NTUT MMS LAB 6

MMS Lab Points

- 函數的傳值呼叫 (call by value)

步驟 1: 在main呼叫cubeByValue之前:

```
int main( void )
{
    int number = 5;
    number = cubeByValue( number );
}
```

number: 5

```
int cubeByValue( int n )
{
    return n * n * n;
}
```

n: undefined

步驟 2: 在cubeByValue接收呼叫之後:

```
int main( void )
{
    int number = 5;
    number = cubeByValue( number );
}
```

number: 5

```
int cubeByValue( int n )
{
    return n * n * n;
}
```

n: 5

NTUT MMS LAB 7

MMS Lab Points

- 函數的傳值呼叫 (call by value)

步驟 3: 在cubeByValue為參數n計算立方值之後，並在cubeByValue返回main之前:

```
int main( void )
{
    int number = 5;
    number = cubeByValue( number );
}
```

number: 5

```
int cubeByValue( int n )
{
    return 125;
}
```

n: 5

步驟 4: 在cubeByValue返回main之後，並在將結果設定給number之前:

```
int main( void )
{
    int number = 5;
    number = cubeByValue( number );
}
```

number: 125

```
int cubeByValue( int n )
{
    return n * n * n;
}
```

n: undefined

NTUT MMS LAB 8

MMS Lab Points

- 函數的傳值呼叫 (call by value)

步驟 5: 在main完成number的設置動作之後:

```
int main( void )
{
    int number = 5;
    125
    125
    number = cubeByValue( number );
}
```

number: 125

```
int cubeByValue( int n )
{
    return n * n * n;
}
```

n: undefined

NTUT MMS LAB 9

MMS Lab Points

- 函數的傳位址呼叫 (call by address)

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void cubeByReference( int *nPtr );
5
6 int main( void )
7 {
8     int number = 5;
9     printf( "The original value of number is %d", number );
10
11     cubeByReference( &number );
12
13     printf( "\nThe new value of number is %d\n", number );
14     system( "pause" );
15     return 0;
16 }
17
18 void cubeByReference( int *nPtr )
19 {
20     *nPtr = *nPtr * *nPtr * *nPtr;
21 }
```

C:\c_code\ch3-87\Debug\ch3-87.exe

The original value of number is 5
The new value of number is 125
請按任意鍵繼續 . . .

NTUT MMS LAB 10

MMS Lab Points

- 函數的傳位址呼叫 (call by address)

步驟 1: 在main呼叫cubeByReference之前:

```
int main( void )
{
    int number = 5;
    cubeByReference( &number );
}
```

number: 5

```
void cubeByReference( int *nPtr )
{
    *nPtr = *nPtr * *nPtr * *nPtr;
}
```

nPtr: undefined

步驟 2: 在呼叫cubeByReference之後，在*nPtr的立方值計算之前:

```
int main( void )
{
    int number = 5;
    cubeByReference( &number );
}
```

number: 5

```
void cubeByReference( int *nPtr )
{
    *nPtr = *nPtr * *nPtr * *nPtr;
}
```

nPtr: call establishes this pointer

NTUT MMS LAB 11

MMS Lab Points

- 函數的傳位址呼叫 (call by address)

步驟 3: 在*nPtr的立方值計算之後，在程式控制權回到main之前:

```
int main( void )
{
    int number = 5;
    cubeByReference( &number );
}
```

number: 125

```
void cubeByReference( int *nPtr )
{
    *nPtr = *nPtr * *nPtr * *nPtr;
}
```

nPtr: called function modifies caller's variable

NTUT MMS LAB 12

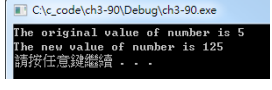
MMS Lab Points

- 函數的傳參考呼叫 (call by reference)
 - C 沒有支援 call by reference, 需要把 main.c 改成 main.cpp

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void cubeByReference( int *nPtr );
5
6 int main( void )
7 {
8     int number = 5;
9     printf("The original value of number is %d", number);
10
11     cubeByReference( number );
12
13     printf("\nThe new value of number is %d\n", number);
14     system("pause");
15     return 0;
16 }
17
18 void cubeByReference(int *nPtr)
19 {
20     *nPtr = *nPtr * *nPtr * *nPtr;
21 }

```



NTUT MMS LAB 1.3

MMS Lab Points

- 氣泡排序 (bubble sort) 使用 call by address
 - 函數呼叫: swap(&array[j], &array[j+1])
 - 函數定義: void swap(int *element1Ptr, int *element2Ptr)

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #define SIZE 10
4
5 void bubbleSort(int * const array, const int size);
6
7 int main( void )
8 {
9     int a[SIZE] = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};
10    int i;
11
12    printf("Data items in original order\n");
13    for (i=0; i<SIZE; i++)
14    {
15        printf("%4d", a[i]);
16    }
17
18    bubbleSort(a, SIZE);
19
20    printf("\nData items in ascending order\n");
21    for (i=0; i<SIZE; i++)
22    {
23        printf("%4d", a[i]);
24    }
25
26    printf("\n");
27    system("pause");
28    return 0;
29 }

```

NTUT MMS LAB 14

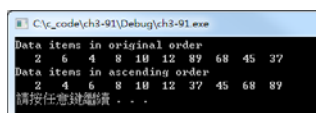
MMS Lab Points

- 氣泡排序 (bubble sort) 使用 call by address
 - 函數呼叫: swap(&array[j], &array[j+1])
 - 函數定義: void swap(int *element1Ptr, int *element2Ptr)

```

32 void bubbleSort(int * const array, const int size)
33 {
34     for (pass=0; pass<size-1; pass++)
35     {
36         for (j=0; j<size-1; j++)
37         {
38             if (array[j] > array[j+1])
39             {
40                 swap( &array[j], &array[j+1]);
41             }
42         }
43     }
44 }
45
46 void swap(int *element1Ptr, int *element2Ptr)
47 {
48     int hold = *element1Ptr;
49     *element1Ptr = *element2Ptr;
50     *element2Ptr = hold;
51 }

```



NTUT MMS LAB 15

MMS Lab Points

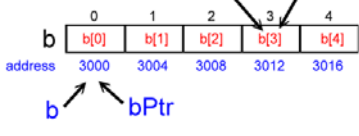
- 整個陣列會自動以傳位址呼叫 (call by address) 來傳遞
- 但個別的陣列元素是純量，如果程式只使用傳值呼叫 (call by value) 來傳遞，無法真正的交換數值
- 陣列的元素，若要真正的交換數值，可以使用傳位址呼叫 (call by address)
 - 函數呼叫: swap(&array[j], &array[j+1])
 - 函數定義: void swap(int *element1Ptr, int *element2Ptr)

	a[0]	a[1]	a[2]	a[3]	a[4]
i=0	5	26	81	7	63
j=1	5	26	81	7	63
j=2	5	26	7	81	63
j=3	5	26	7	63	81

NTUT MMS LAB 16

MMS Lab Points

- 指標和陣列
 - 陣列的名稱可以想像是一個指向陣列第一個元素的指標
 - EX: b[5];
 - bPtr = b; // bPtr 指定陣列b的第一個元素位址
 - bPtr = &b[0]; // bPtr 指定陣列b的第一個元素位址
 - *(bPtr+3) // 代表b[3]裡面的元素



NTUT MMS LAB 17

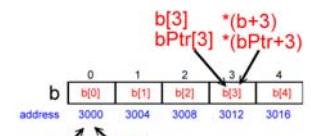
MMS Lab Points

- 指標和陣列

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(void)
5 {
6     int b[] = {10, 20, 30, 40};
7     int *bPtr = b;
8     int i;
9     int offset;
10
11     printf("Array b printed with:\narray subscript notation\n");
12     for (i=0; i<4; i++)
13     {
14         printf("b[%d] = %d\n", i, b[i]);
15     }
16
17     printf("\nPointer/offset notation where\n"
18           "the pointer is the array name\n");
19     for (offset=0; offset<4; offset++)
20     {
21         printf("*(b + %d) = %d\n", offset, *(b+offset));
22     }
23 }

```



NTUT MMS LAB 18

MMS Lab Points

- 指標和陣列

```

23 printf("\nPointer subscript notation\n");
24 for (i=0; i<4; i++)
25 {
26     printf("bPtr[%d] = %d\n", i, bPtr[i]);
27 }
28
29 printf("\nPointer/offset notation\n");
30 for (offset=0; offset<4; offset++)
31 {
32     printf("*(bPtr + %d) = %d\n", offset, *(bPtr+offset));
33 }
34
35 system("pause");
36 return 0;
37 }

```

NTUT MMS LAB

19

MMS Lab Points

- 指標陣列 (array of pointer)

```

const char *suit[ 4 ] = { "Hearts", "Diamonds", "Clubs", "Spades" };

```

圖 7.22 suit 陣列的圖形表示

NTUT MMS LAB

20

MMS Lab Points

- 模擬發牌程式-指標陣列 (array of pointer)
 - 首先設定deck[4][13]初始值都是0, 尚未設定順序
 - int deck[4][13]={0};
 - 設定deck[4][13]出牌順序
 - shuffle(deck);
 - 檢查deck[4][13]出牌順序, 由1到52列印花色和大小
 - deal(deck);

NTUT MMS LAB

21

MMS Lab Points

- 模擬發牌程式-指標陣列 (array of pointer)


```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 void shuffle(int wDeck[][13]);
6 void deal(const int wDeck[][13], const char *wFace[],
7           const char *wSuit[]);
8
9 int main(void)
10 {
11     const char *suit[4] = {"Hearts", "Diamonds", "Clubs", "Spades"};
12
13     const char *face[13] =
14     {"Ace", "Deuce", "Three", "Four",
15      "Five", "Six", "Seven", "Eight",
16      "Nine", "Ten", "Jack", "Queen", "King"};
17
18     int deck[4][13] = {0};
19
20     srand(time(0));
21
22     shuffle(deck);
23     deal(deck, face, suit);
24     system("pause");
25     return 0;
26 }

```

NTUT MMS LAB

22

MMS Lab Points

- 模擬發牌程式-設定deck[4][13]出牌順序
 - while (wDeck[row][column] != 0) //代表尚未給定順序牌號
 - wDeck[row][column] = card; //給定順序牌號,由小到大
 - wDeck[row][column] = 1;
 -
 - wDeck[row][column] = 52;

```

27 void shuffle(int wDeck[][13])
28 {
29     int row;
30     int column;
31     int card;
32
33     for (card=1; card<=52; card++)
34     {
35         do
36         {
37             row = rand() % 4;
38             column = rand() % 13;
39             while (wDeck[row][column] != 0);
40             wDeck[row][column] = card;
41         }
42     }
43 }
44

```

NTUT MMS LAB

23

MMS Lab Points

- 模擬發牌程式-檢查deck[4][13]出牌順序, 由1到52列印花色和大小


```

45
46 void deal(const int wDeck[][13], const char *wFace[],
47           const char *wSuit[])
48 {
49     int card;
50     int row;
51     int column;
52
53     for (card=1; card<=52; card++)
54     {
55         for (row=0; row<3; row++)
56         {
57             for (column=0; column<12; column++)
58             {
59                 if (wDeck[row][column] == card)
60                 {
61                     printf("%5s of %-8s%c", wFace[column], wSuit[row],
62                           card % 2 == 0 ? '\n' : '\t');
63                 }
64             }
65         }
66     }
67 }

```

NTUT MMS LAB

24

• 模擬發牌程式

```
C:\Users\Andy\Desktop\ch3-100\Debug\ch3-100.exe
Six of Spades      Eight of Diamonds
Seven of Hearts    Four of Spades
Ace of Clubs       Three of Spades
Seven of Spades    King of Spades
Ten of Diamonds    Three of Hearts
Ace of Diamonds    Nine of Hearts
Ace of Spades      Ace of Hearts
Four of Clubs      Deuce of Diamonds
Seven of Clubs     Nine of Clubs
Four of Diamonds   Six of Diamonds
Deuce of Clubs     Five of Hearts
Six of Clubs       Three of Clubs
King of Clubs      King of Diamonds
Five of Clubs      Eight of Spades
Nine of Diamonds   Jack of Clubs
Ten of Hearts      Jack of Hearts
Five of Diamonds   Three of Diamonds
Queen of Spades    King of Hearts
Seven of Diamonds  Nine of Spades
Six of Hearts      Jack of Spades
Jack of Diamonds   Eight of Clubs
Queen of Diamonds  Ten of Clubs
Queen of Clubs     Deuce of Spades
Deuce of Hearts    Eight of Hearts
```