

## 程式設計第七章



真理大學資工系 洪麗玲  
llhung@mail.au.edu.tw

### 7.1 簡介



- **指標 (pointer)** 是C程式語言最強大的功能之一，我們將在本章中討論。指標能讓程式模擬傳參考讓函式之間能互相傳遞，以及產生和操作動態的資料結構，亦即在執行時期會增大和減小的資料結構，如鏈結串列 (linked lists)、佇列、堆疊和樹。

## 7.2 指標變數的定義及初始值設定



- 指標是代表**記憶體位址**的**變數**。通常一個變數都會存放某個特定的數值。而指標所存放的卻是某個變數的位址。在這種認知下，我們可以說一個變數名稱**直接 (directly) 指到**一個值，而一個指標則**間接 (indirectly) 指到**這個值。透過指標來**參考**某個值稱為**間接 (indirection)**。

WATSE



### ❖ 宣告指標

- 指標和其他變數一樣，必須在使用之前進行定義。底下的宣告

```
int *countPtr, count;
```

count



count 直接參考一個  
值為7的變數

countPtr



count



指標countPtr 間接參  
考一個值為7的變數

- 指定變數**countPtr**的型別為**int \*** (也就是指向整數的指標)。

WATSE



### ❖ 對指標初始化及設定值

- 指標應該在定義時初始化，可能將初始值設定成0，**NULL**或某個位址。**NULL**的指標不指向任何東西。**NULL**是定義在<stddef.h> 標頭檔中的符號常數。將指標初始化為0與初始化為**NULL**是一樣的，但是使用**NULL**較佳。當0設定給指標時，它會先轉換成適當型別的指標。0是唯一可以直接設定給指標變數的整數。

WATSE

## 7.3 指標運算子



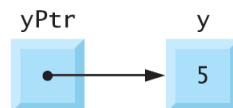
### ❖ &運算子，或稱為取址運算子 (address operator)

- 是一個會傳回運算元位址的一元運算子舉例來說，我們假設以下的定義

```
int y = 5;  
int *yPtr;
```

- 則以下的敘述式

```
yPtr = &y;
```



WATSE



### ❖ 指標在記憶體中的形式

- 圖7.3展示了指標在記憶體中的樣子，假設整數變數 $y$ 儲存在位址600000，而指標變數 $yPtr$ 儲存在位址500000。取址運算子的運算元必須是個變數；取址運算子不能應用到常數或運算式。



圖7.3  $y$ 和 $yPtr$ 的記憶體表示圖

WATSE



### ❖ 間接運算子\*

- \*運算子通常稱為**間接運算子 (indirection operator)** 或**反參考運算子 (dereferencing operator)**，它會傳回其運算元 (即指標) 所指向的物件的數值。例如，下面的敘述式

```
printf( "%d", *yPtr );
```

- 將會印出變數 $y$ 的值，也就是5。以這種方式來使用\*就稱為**反參考指標 (dereferencing a pointer)**。

WATSE

## ❖ 指標運算子 & 和 \* 的使用方法

### ■ 圖7.4說明 & 以及 \* 指標運算子的使用方法



```

1 // Fig. 7.4: fig07_04.c
2 // Using the & and * pointer operators.
3 #include <stdio.h>
4
5 int main( void )
6 {
7     int a; // a is an integer
8     int *aPtr; // aPtr is a pointer to an integer
9
10    a = 7;
11    aPtr = &a; // set aPtr to the address of a
12
13    printf( "The address of a is %p"
14           "\nThe value of aPtr is %p", &a, aPtr );
15
16    printf( "\n\nThe value of a is %d"
17           "\nThe value of *aPtr is %d", a, *aPtr );
18
19    printf( "\n\nShowing that * and & are complements of "
20           "each other\n&*aPtr = %p"
21           "\n*&aPtr = %p\n", &*aPtr, *&aPtr );
22 } // end main

```

```

7     int a; // a is an integer
8     int *aPtr; // aPtr is a pointer to an integer
9
10    a = 7;
11    aPtr = &a; // set aPtr to the address of a
12
13    printf( "The address of a is %p"
14           "\nThe value of aPtr is %p", &a, aPtr );
15
16    printf( "\n\nThe value of a is %d"
17           "\nThe value of *aPtr is %d", a, *aPtr );
18
19    printf( "\n\nShowing that * and & are complements of "
20           "each other\n&*aPtr = %p"
21           "\n*&aPtr = %p\n", &*aPtr, *&aPtr );
22 } // end main

```

```

The address of a is 0028FEC0
The value of aPtr is 0028FEC0

```

```


The value of a is 7
The value of *aPtr is 7

```

```

Showing that * and & are complements of each other
&*aPtr = 0028FEC0
*&aPtr = 0028FEC0

```



| 運算子                             | 結合性  | 形式     |
|---------------------------------|------|--------|
| () [] ++ (postfix) -- (postfix) | 由左至右 | 最高     |
| + - ++ -- ! * & (type)          | 由右至左 | 一元     |
| * / %                           | 由左至右 | 乘法     |
| + -                             | 由左至右 | 加法     |
| < <= > >=                       | 由左至右 | 關係     |
| == !=                           | 由左至右 | 相等     |
| &&                              | 由左至右 | 邏輯 AND |
|                                 | 由左至右 | 邏輯 OR  |
| ?:                              | 由右至左 | 條件     |
| = += -= *= /= %=                | 由右至左 | 指定     |
| ,                               | 由左至右 | 逗號     |

圖7.5 運算子的運算優先順序與結合性

WATSE

## 7.4 傳參考呼叫



### ❖ 傳值呼叫

```
main{... fun1(int t)...}
```

```
int fun1(int d){...}
```

### ❖ 利用指標和間接運算子來進行傳參考呼叫：

若傳給某個函式的引數應該要被更改的話，則傳遞此引數的位址給函式。以變數之前加上 **&** 來加以達成。當傳遞變數的位址給函式時，函式可以利用間接運算子 (**\***) 來接收並更改位於呼叫者記憶體內的數值。

```
main{... fun1(&t)...}
```

```
int fun1(int *d){...}
```

WATSE

```

1 // Fig. 7.6: fig07_06.c
2 // Cube a variable using pass-by-value.
3 #include <stdio.h>
4
5 int cubeByValue( int n ); // prototype
6
7 int main( void )
8 {
9     int number = 5; // initialize number
10
11     printf( "The original value of number is %d", number );
12
13     // pass number by value to cubeByValue
14     number = cubeByValue( number );
15
16     printf( "\nThe new value of number is %d\n", number );
17 } // end main
18
19 // calculate and return cube of integer argument
20 int cubeByValue( int n )
21 {
22     return n * n * n; // cube local variable n and return result
23 } // end function cubeByValue

```

The original value of number is 5  
The new value of number is 125

圖7.6 使用傳值呼叫來將某變數設定為它的立方值



```

1 // Fig. 7.7: fig07_07.c
2 // Cube a variable using pass-by-reference with a pointer argument.
3
4 #include <stdio.h>
5
6 void cubeByReference( int *nPtr ); // function prototype
7
8 int main( void )
9 {
10     int number = 5; // initialize number
11
12     printf( "The original value of number is %d", number );
13
14     // pass address of number to cubeByReference
15     cubeByReference( &number );
16
17     printf( "\nThe new value of number is %d\n", number );
18 } // end main
19
20 // calculate cube of *nPtr; actually modifies number in main
21 void cubeByReference( int *nPtr )
22 {
23     *nPtr = *nPtr * *nPtr * *nPtr; // cube *nPtr
24 } // end function cubeByReference

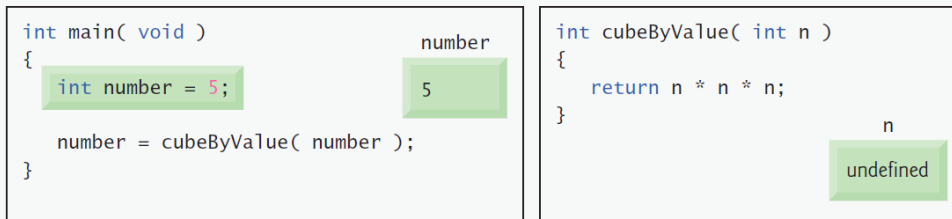
```

The original value of number is 5  
The new value of number is 125

圖7.7 使用傳參考呼叫，以指標引數將某變數設定為其立方值



步驟 1: 在main呼叫cubeByValue之前:



步驟 2: 在cubeByValue接收呼叫之後:

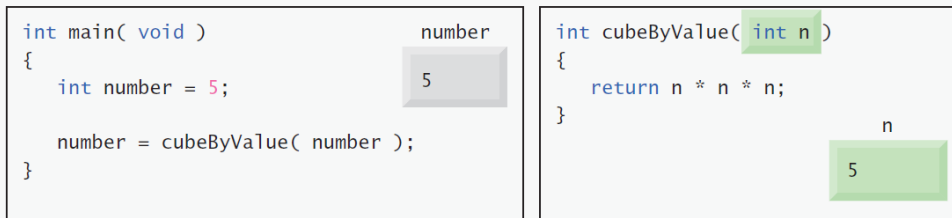
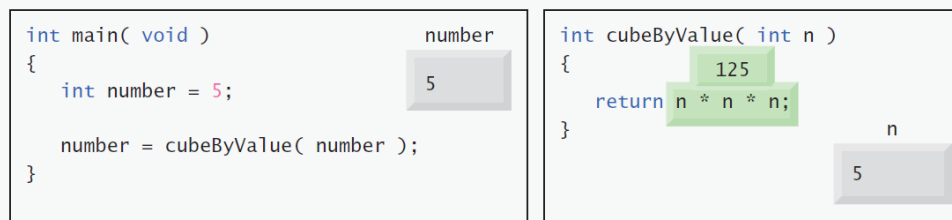


圖7.8 典型傳值呼叫的分析(1/3)



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



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

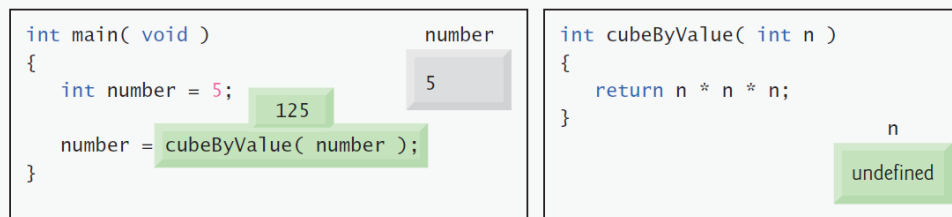


圖7.8 典型傳值呼叫的分析(2/3)







步驟 5: 在main完成number的設值動作之後:

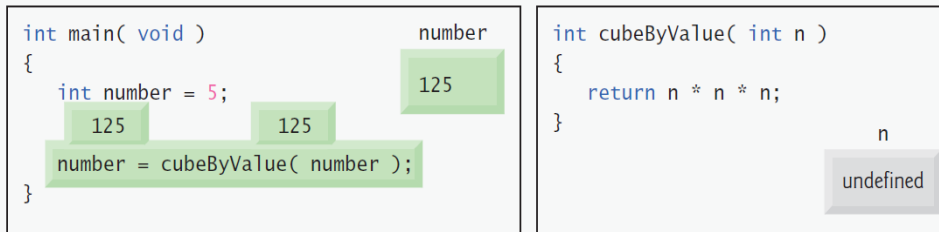
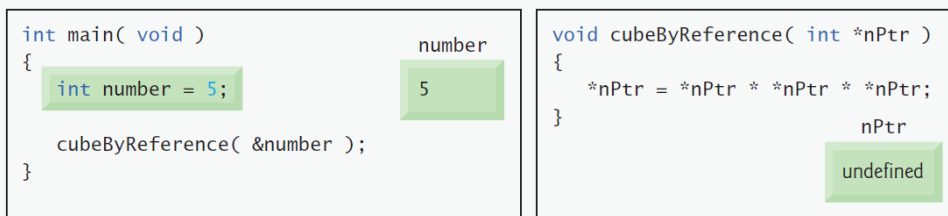


圖7.8 典型傳值呼叫的分析(3/3)



步驟 1: 在main呼叫cubeByReference之前:



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

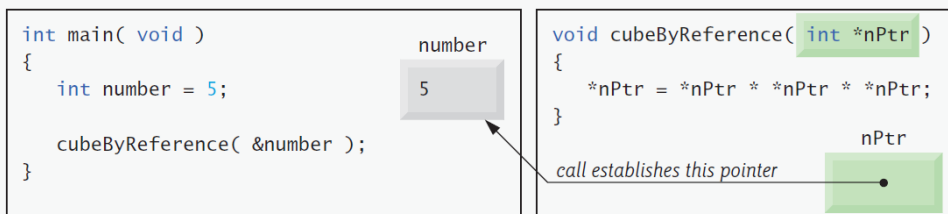


圖7.9 使用指標引數的典型傳參考的分析(1/2)





步驟 3: 在 \*nPtr 的立方值計算之後，在程式控制權回到 main 之前:

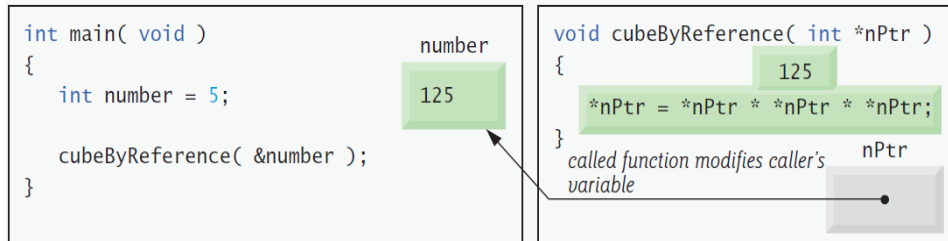


圖7.9 使用指標引數的典型傳參考的分析(2/2)

WATSE

## 作業



❖ 題目: 小於N的完整數有哪些?

❖ **int main(void):**

- 要求使用者輸入數值N，找出比N小的最大完整數傳回 FindMaxComplete(請同時使用傳值與傳參考方式)

❖ **int FindMaxComplete(???, ???) :**

- 若有找到符合的完整數回傳值是1
- 若沒找到符合的完整數回傳值是0

WATSE