程式設計
(Programming)

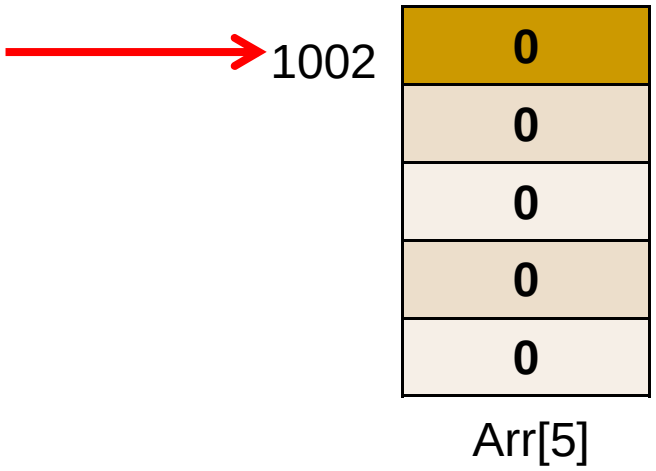
真理大學 資訊工程系 吳汶涓老師

CH07
指標(pointer)

本章綱要

7.1 簡介

- 指標 (pointer)
  - 是C語言最強大的功能之一，但難以駕馭
  - 可模擬傳參考呼叫：呼叫時，只告知資料的記憶體位置
  - 與陣列和字串有密切關係



7.2 指標變數的定義及初始值設定

- 指標變數
  - 其值代表某個記憶體位址
  - 一般變數會存放某個特定的數值(直接參考)
  - 指標所存放的是某個變數的位址(間接參考) ← 透過指標參考某個值

- 定義: int *countPtr;
  - int *: 宣告指向一個整數的指標
  - int *countPtr, count=7;

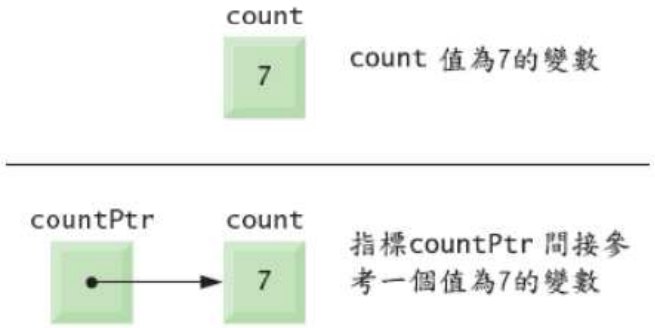


圖 7.1 直接和間接參考一個變數

- 初始值設定
  - 指標設定初始值爲0、NULL或一個位址
    - 0或NULL: 表示不指向任何東西 → int *countPtr = NULL;

常見的程式設計錯誤 7.1

星號 (*) 的效用並不會作用到宣告內的所有變數名稱。每一個指標在宣告時都必須在它的名稱之前加上*，也就是如果要將 xPtr 和 yPtr 宣告成 int 指標，必須寫成 int *xPtr, *yPtr;。

int *xPtr, yPtr; (X)
int *xPtr, *yPtr; (O)

常見的程式設計錯誤 7.2

將 ptr 這三個字母放到指標變數的名稱中，可清楚表示這個變數是一個指標。

課本pp. 7-3

7.3 指標運算子

- 取址運算子 (&)
  - 傳回運算元的地址

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

將y的地址指定給指標變數 yPtr

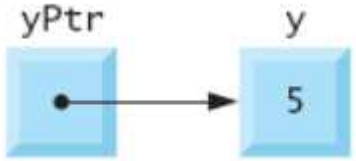


圖 7.2　指標的記憶體表示圖

- 間接/反參考運算子(*)
  - 傳回運算元(指標)所指向的值

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

  - 亦可用來給定值 ➔ `*yPtr = 7;`

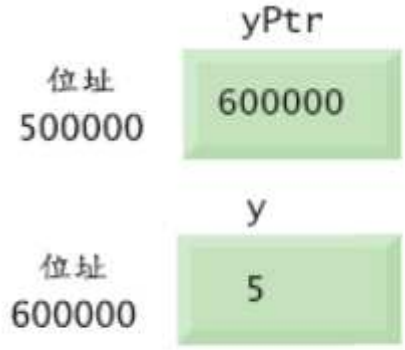


圖 7.3　y 和 yPtr 的記憶體

- &和*是互補的
  - 彼此互相抵消

常見的程式設計錯誤 7.3

反參考一個沒有正確初始化或沒有指定爲指向記憶體特定位置的指標將會導致錯誤。這會導致致命的執行期錯誤，或將不小心更改到重要的資料，並且程式會完成它的執行，但卻得到錯誤的結果。

```
int  y = 5, *yPtr;
*yPtr = 7;
```

課本pp. 7-5

```c
/* Fig. 7.4: fig07_04.c */
#include <stdio.h>
int main( void )
{
   int a;
   int *aPtr; /* aPtr is a pointer to an integer */

   a = 7;
   aPtr = &a;

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

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

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

一樣，因爲aPtr指向a

&*互相抵銷，所以一樣

```
The address of a is 0012FF7C
The value of aPtr is 0012FF7C

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

Showing that * and & are complements of each other.
&*aPtr = 0012FF7C
*&aPtr = 0012FF7C
```

圖 7.4　使用 & 和 * 指標運算子

課本pp. 7-5

練習

- 撰寫每一項的指令敘述式。假設 long整數變數value1和value2已宣告，且value1的初始值爲200000。

a) 定義一個指向long型別變數的指標 v1Ptr

```
long  *v1Ptr;
```

b) 將變數 value1 的位址指定給指標變數 v1Ptr

```
v1Ptr = &value1;
```

c) 印出v1Ptr所指變數的值

```
printf("%ld", *v1Ptr);
```

d) 將v1Ptr所指的值指定給變數 value2

```
value2 = *v1Ptr;
```

e) 印出value1的位址 `printf("%p", &value1);`

Q: v1Ptr參考的值是否與value2的數值一樣呢??

Q: v1Ptr參考的位址是否與value1的位址一樣呢?

課本pp. 7-47

7.4 傳參考呼叫

- 函式參數可用傳值呼叫、傳參考呼叫(call by reference)
- 傳參考呼叫: 在函式參數上使用指標
  - 利用 & 運算子傳遞資料位址
  - 可以改變記憶體中的實際位置
  - 傳遞陣列時不需使用 &，因爲陣列名稱就是一個指標

  - 範例：

函式

```
void doubleMulti(int *ptr){
        *ptr = 2* (*ptr);
}
```

主程式

```
int main(void){
        int x = 5;
        doubleMulti(&x);
        printf("%d", x);
}
```

值爲多少?

- 範例：計算三次方值 (函式參數使用**call by value**)

```
1  /* Fig. 7.6: fig07_06.c */
3  #include <stdio.h>
4
5  int cubeByValue( int n );
6
7  int main( void )
8  {
9     int number = 5;
10
11    printf( "The original value of number is %d", number );
14    number = cubeByValue( number );
15
16    printf( "\nThe new value of number is %d\n", number );
17    return 0;
18 }
19
21 int cubeByValue( int n )
22 {
23    return n * n * n;
24 }
```

```
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

課本pp. 7-7

- 範例：計算三次方值(函式參數使用**call by reference**)

```
1  /* Fig. 7.7: fig07_07.c */
4  #include <stdio.h>
5
6  void cubeByReference( int *nPtr );
7
8  int main( void )
9  {
10    int number = 5;
11
12    printf( "The original value of number is %d", number );
15    cubeByReference( &number );
16
17    printf( "\nThe new value of number is %d\n", number );
18    return 0;
19 }
20
22 void cubeByReference( int *nPtr )
23 {
24    *nPtr = *nPtr * *nPtr * *nPtr;
25 }
```

函式內有個指標參數

將變數的位址傳遞給函式

```
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:

```
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

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 n * n * n;
}
```

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 );
}
```

125

number: 5

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

n: undefined

圖 7.8 典型傳值呼叫的分析

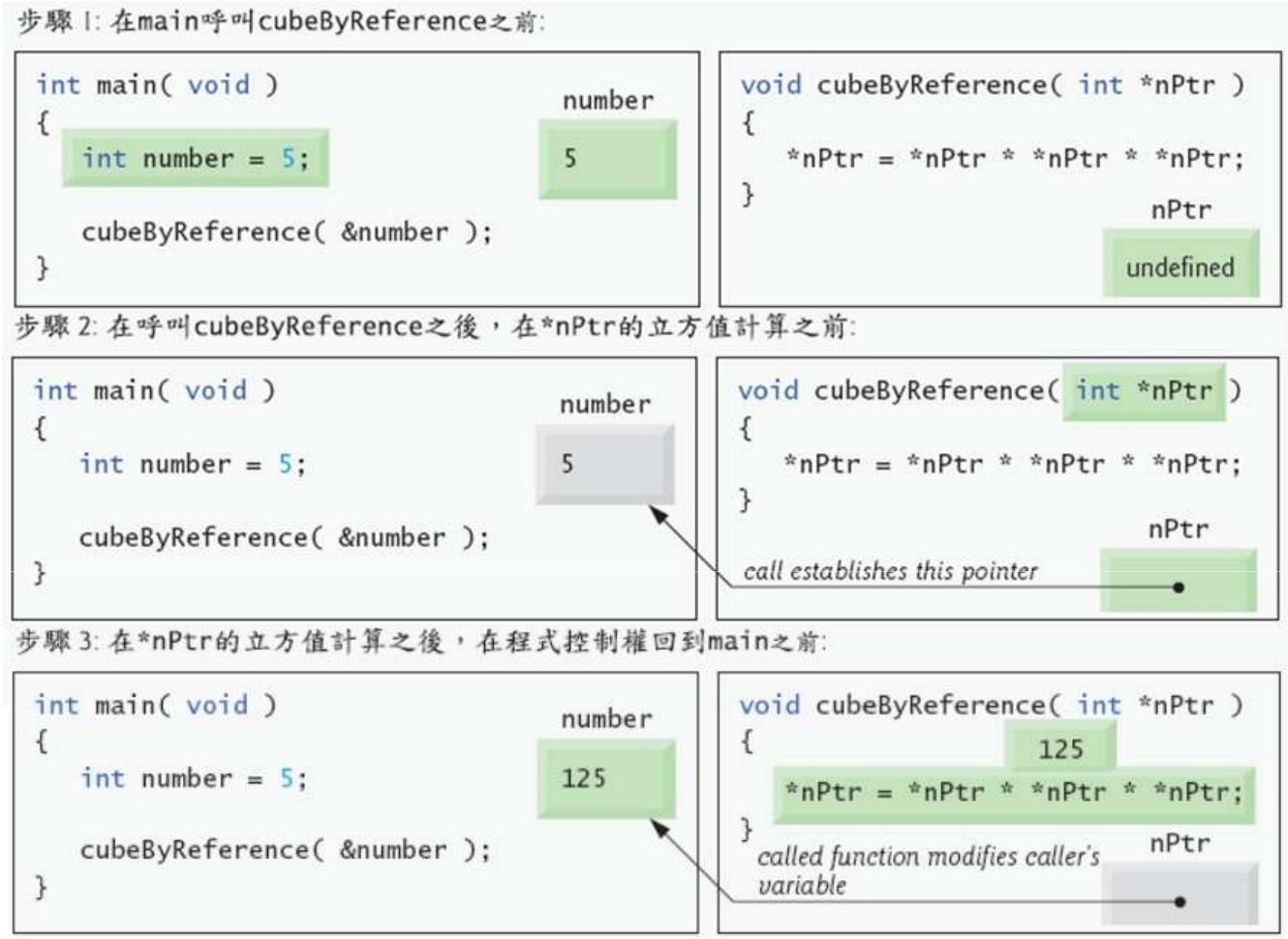


圖 7.9　使用指標引數的典型傳參考呼叫的分析

練習

- 請撰寫一程式，由使用者選擇攝氏轉華氏、華氏轉攝氏之功能，兩者分別使用下列函式呼叫。
  - 攝氏轉華氏:
    void CtoF(float *Ptr)
  - 華氏轉攝氏:
    void FtoC(float *Ptr)

```
select 1 or 2:
1. 攝氏轉華氏
2. 華氏轉攝氏
ans: 1
temp:33.5
temp= 92.30
請按任意鍵繼續 . . .
```

使用者輸入的溫度有小數值!