第七章講授重點

指標(pointer)

講授重點

1. 指標定義與使用
2. 傳參考呼叫
3. 傳參考呼叫的氣泡搜尋法

1. 指標定義與使用

- 指標 (pointer) 是C程式語言最強大的功能之一。
- 指標讓程式傳參考呼叫，及產生和操作動態的資料結構，即增大和減小的資料結構，如鏈結串列 (linked lists)、佇列、堆疊和樹。
- 第十章討論使用指標的結構。
- 第十二章則介紹動態記憶體管理 (dynamic memory management) 技術，及產生和使用動態資料結構的例子。

1. 指標定義與使用(續)

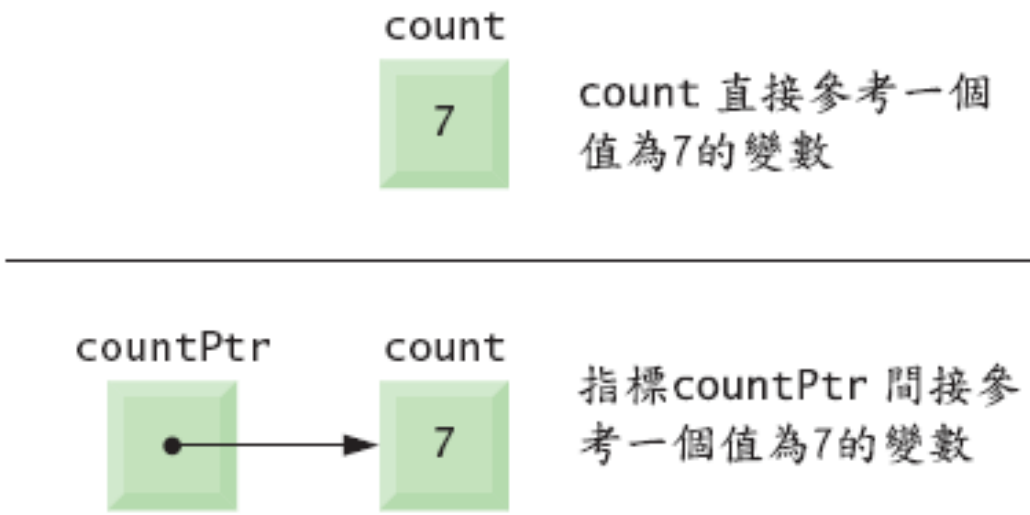


圖 7.1　直接和間接參考一個變數

1. 指標定義與使用(續)

- 變數都會存放某個特定的數值。
- 變數名稱直接 (directly) 參考一個值
- 指標存放是變數的位址。
- 指標則間接 (indirectly) 參考這個值。

1. 指標定義與使用(續)

- 指標和其他變數一樣，使用前定義。
- 定義

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

countPtr的資料型態爲整數指標
count的資料型態爲整數

1. 指標定義與使用(續)

- **&** 運算子，稱爲取址運算子 (address operator)，是傳回變數(或稱運算元)所在位址的運算子
- 例如，

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

敘述式

```
yPtr = &y;
```

將變數y的位址傳給指標變數yPtr。

- 這時，指標變數yPtr就「指向」變數y。

1. 指標定義與使用(續)

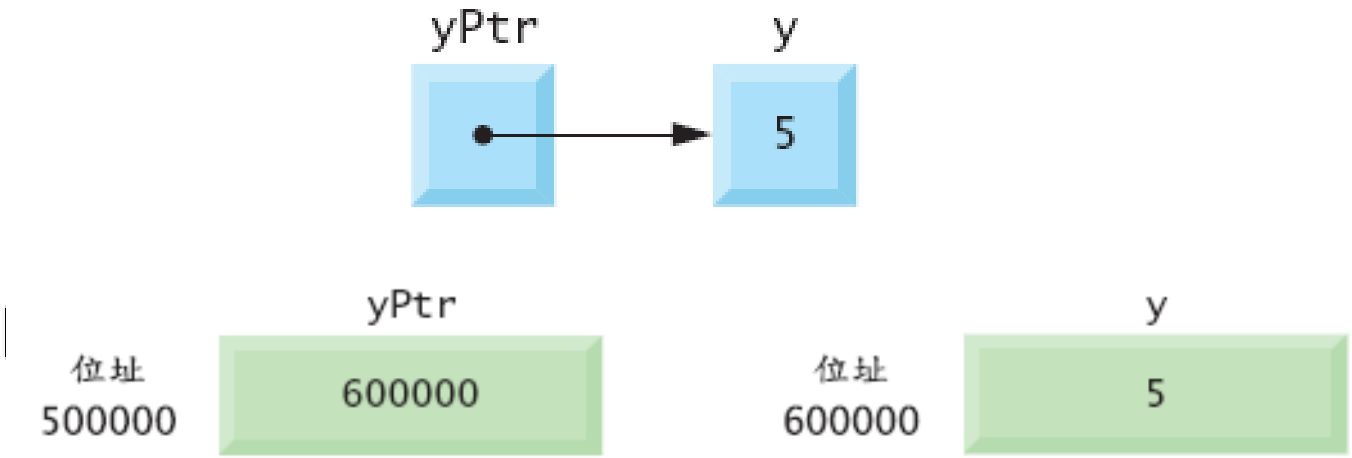


圖 7.3　y 和 yPtr 的記憶體表示圖

- 取址運算子的運算元必須是變數；
- 取址運算子不能應用到常數、運算式、或宣告爲register的變數。

1. 指標定義與使用(續)

- * 運算子稱爲間接運算子 (indirection operator) 或反參考運算子 (dereferencing operator)，傳回其運算元 (即指標) 所指向物件(或變數)的數值。
- 例如
  - `printf( "%d", *yPtr );`

  將印出變數y的值，也就是5。

例題一

```
1  /* Fig. 7.4: fig07_04.c
2     Using the & and * 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; /* aPtr set to 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    return 0; /* indicates successful termination */
23 } /* end main */
```

Fig. 7.4 | Using the & and * pointer operators. (Part 1 of 2.)

例題一(續)

```
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
```

Fig. 7.4 | Using the & and * pointer operators. (Part 2 of 2.)

2. 傳參考呼叫

- 若傳給函式的引數(或變數)需回傳至主程式，則需傳遞此引數(或變數)的位址給函式。
- 故需在變數之前加上&
- 如 scanf(“%d”, &height)
- 第六章中提過，陣列傳遞不用加上&運算子，因爲C自動傳遞陣列在記憶體中的起始位址。
- 如 scanf(“%s %d”, name[i], &height[i])
- 當傳遞變數的位址給函式時，函式可以利用間接運算子(*)更改位於呼叫者記憶體內的**數值**。

2. 參考呼叫(續)

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

圖 7.9　使用指標引數的典型傳參考呼叫的分析

2. 參考呼叫(續)

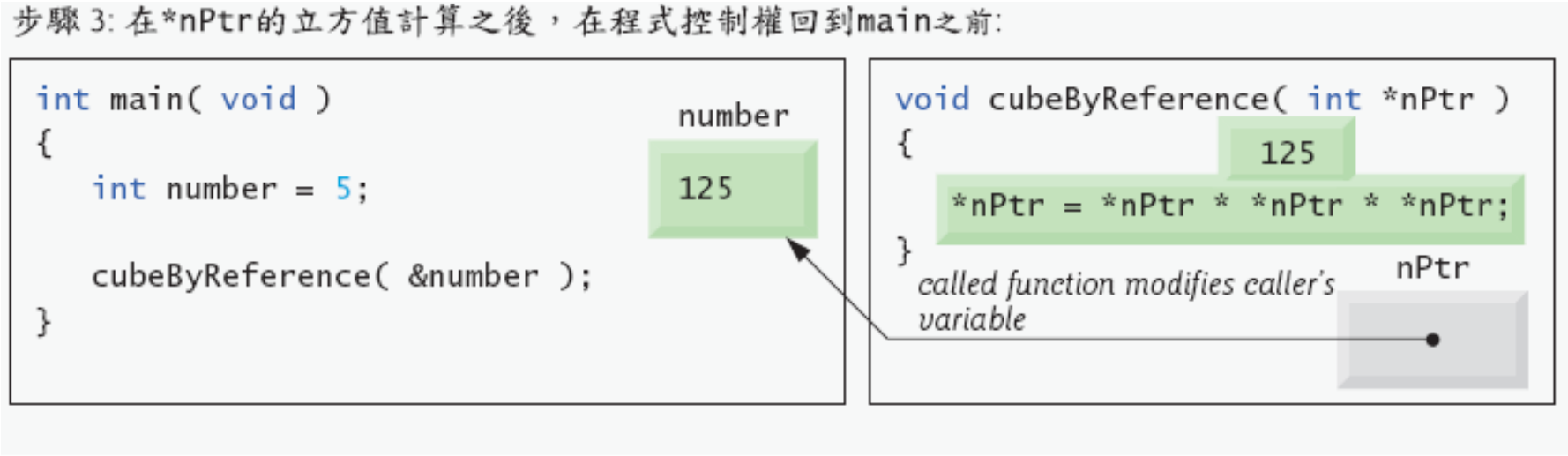


圖 7.9　使用指標引數的典型傳參考呼叫的分析

例題二

```
1  /* Fig. 7.7: fig07_07.c
2     Cube a variable using call-by-reference with a pointer argument */
3
4  #include <stdio.h>
5
6  void cubeByReference( int *nPtr ); /* 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    return 0; /* indicates successful termination */
19 } /* end main */
20
```

Fig. 7.7 | Cube a variable using call-by-reference with a pointer argument. (Part 1 of 2.)

例題二(續)

```
21  /* calculate cube of *nPtr; modifies variable number in main */
22  void cubeByReference( int *nPtr )
23  {
24     *nPtr = *nPtr * *nPtr * *nPtr; /* cube *nPtr */
25  } /* end function cubeByReference */
```

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

Fig. 7.7 | Cube a variable using call-by-reference with a pointer argument. (Part 2 of 2.)

例題三

t1(area, fuct, ptr).c

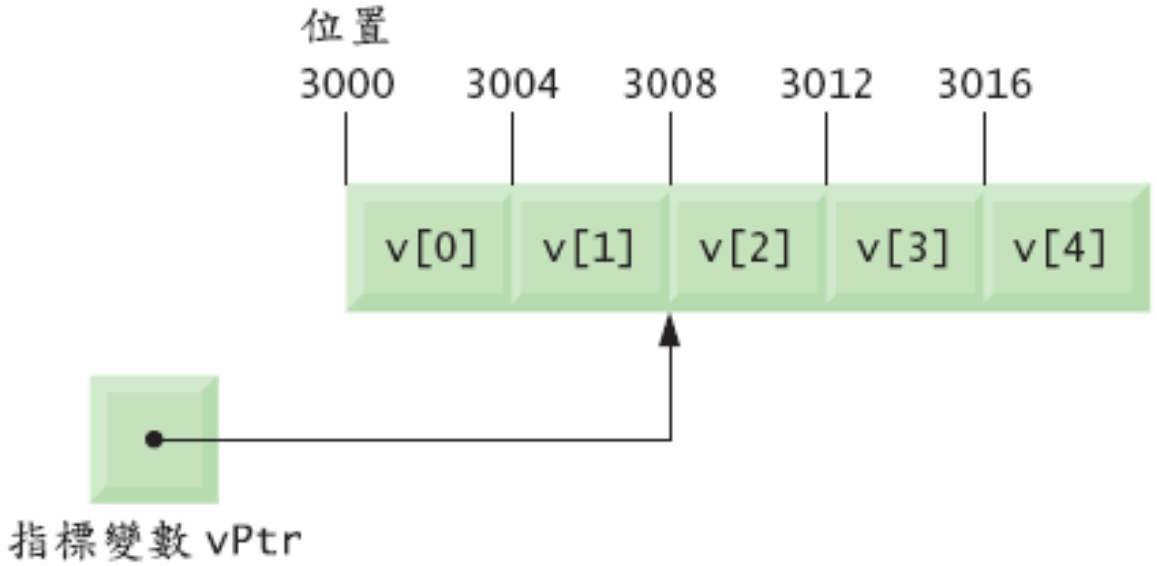
```c
// t1 : to find the area of rectangle
#include <stdio.h>

float area(float x, float y );
float AREA(float x, float y, float *z);

int main( void )
{
   float a,b,c,z;
   printf( "Enter two edges: " );
   scanf( "%f,%f", &a, &b );
   c = area(a,b);
   AREA(a,b,&z);
   printf("c=%f, z=%f", c, z );
   return 0;
}

float area(float x, float y)
{
   float w;
   w = x*y;
   return w;
}


float AREA(float x, float y, float *z)
{
   *z = x*y;
}
```

```
G:\Documents and Settings\hu\桌面\C\1\ch07\t1(are
Enter two edges: 10, 3
c=30.000000, z=30.000000
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

3. 傳參考呼叫的氣泡搜尋法

- 假設陣列int v[5]已經定義過，它的第一個元素(即v[0])在記憶體位置爲3000。
- 假設指標vPtr設定成指向v[0]，即 **vPtr = &v[0]**， vPtr的值爲3000。
- **vPtr += 2 (或 vPtr = vPtr+2);** 將會產生3008 (3000 + 2 * 4)，假設整數爲4個位元組。即 **vPtr = &V[2]**

3. 傳參考呼叫的氣泡搜尋法(續)



原指令 `vPtr = &V[0] ;`
當指令 `vPtr += 2` 或 `vPtr = vPtr+2)` 或 `vPtr = &V[2]`被執行後

3. 傳參考呼叫的氣泡搜尋法(續)

- 取得陣列b中第一個元素的位址，

```
bPtr = &b[ 0 ];
```

- 陣列元素b[3]之數值可用指標運算式

```
*( bPtr + 3 )
```

因爲 `bPtr + 3 = &b[3]`

- 其中3是指由這個指標開始算起的位移值(offset)。

例題四

[*] t4(name, height, ptr _bubble sorting).c

```
 4  int input_data(char N[10][5], int H[], char *NP[], int *HP[]);
 5  void bubble_sort(char *NP[], int *HP[], int T);
 6
 7  int main( void )
 8  {
 9      char name[10][5]={0}; /* 預定十位名字，每個名字僅有4個字元*/
10      int height[10]={0}; /* 預定十個身高 */
11      int i,j, total = 0;
12      char *NamePtr[10]={0}; /*預定十位名字的指標*/
13      int *heightPtr[10]={0};
14
15      total = input_data(name, height, NamePtr, heightPtr);
16
17      printf( "\n" );
18
19      for ( i = 0; i < total ; i++ ) {
20          printf("name = %s, height = %d\n", name[i], height[i]);
21          printf("name = ");
22          j = 0;
23          while( *(NamePtr[i]+j) != '\0'  ){ //使用指標印出名字與身高
24                printf("%c", *(NamePtr[i]+j) );
25                j++;
26          }
27          printf(", height = %d\n", *heightPtr[i]);
28          printf("\n");
29      }
```

例題四(續)

```
31      bubble_sort(NamePtr, heightPtr, total);  //使用指標印完成氣泡排序
32
33      printf("\n --- after bubble sorting ----\n");
34      for ( i = 0; i < total ; i++ ) {
35          j = 0;
36          while( *(NamePtr[i]+j) != '\0'  ){
37                printf("%c", *(NamePtr[i]+j) );
38                j++;
39          }
40          printf(", height = %d\n", *heightPtr[i]);
41      }
42
43      printf( "\n" );
44      return 0;
45  }
```

例題四(續)

```
47  int input_data(char N[10][5], int H[], char *NP[], int *HP[])
48  {
49      int i, I = 0;
50
51      for(i=0; i<10; i++){
52          printf("Enter name and height: ");
53          scanf( "%s %d", N[i], &H[i] ); /* input name and height */
54          if ( H[i] <= 0 ) break;
55          else {
56           I++;
57           NP[i] = &N[i][0];
58           HP[i] = &H[i];
59          }
60          printf("\n");
61      }
62
63      return I;
64
65  }
```

例題四(續)

```
67  void bubble_sort(char *NP[], int *HP[], int T)
68  {
69      int i, j, k,  *hold;
70      char *tmp;
71      for(j=0; j<T; j++){
72        for(i=0; i<T-1; i++ ){
73          if ( *HP[ i ] > *HP[ i + 1 ] ) {
74              hold = HP[ i ];     // 身高轉移
75              HP[ i ] = HP[ i + 1 ];
76              HP[ i + 1 ] = hold;
77              tmp = NP[i];    //姓名轉移
78              NP[ i ] = NP[ i + 1 ];
79              NP[ i + 1 ] = tmp;
80          }
81       }
82    }
83  }
```

```
G:\Documents and Settings\hu\桌面\C1\ch
Enter name and height: a1 100

Enter name and height: b2 200

Enter name and height: c3 80

Enter name and height: d4 150

Enter name and height: 0 0

name = a1, height = 100
name = a1, height = 100
name = b2, height = 200
name = b2, height = 200
name = c3, height = 80
name = c3, height = 80
name = d4, height = 150
name = d4, height = 150

 --- after bubble sorting ----
name = c3, height = 80
name = a1, height = 100
name = d4, height = 150
name = b2, height = 200
```

作業

- 請做作業7_3，請使用指標氣泡搜尋函式(如例題四或所附檔案t4.c)。
- 某家皮鞋現有皮鞋廠牌、型號與價格如下：
- (1)請依價格排序並印出
- (2)請依型號排序並印出

廠牌	型號	價格
A	1	1000
A	3	1200
A	5	1400
B	1	900
B	3	1100
B	5	1300