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

7.5 const修飾詞在指標上的使用

- const 用來指定某個變數的值**不應該進行更改**
  - 若嘗試更改 const 變數，會造成**錯誤訊息**
  - const 用在函式**傳參考呼叫的參數**上有四種方式
    - 指向非常數資料的非常數指標 **(如: char *sPtr)**
    - 指向非常數資料的　常數指標 (如: char *const sPtr)
    - 指向　常數資料的非常數指標 **(如: const char *sPtr)**
    - 指向　常數資料的　常數指標 (如: const char *const sPtr)

課本pp. 7-12

- 指向非常數資料的非常數指標

```
5  #include <stdio.h>
6  #include <ctype.h>
7
8  void convertToUppercase( char *sPtr );
10 int main( void )
11 {
12    char string[] = "characters and $32.98";
14    printf( "The string before conversion is: %s", string );
15    convertToUppercase( string );
16    printf( "\nThe string after conversion is: %s\n", string );
18    return 0;
20 }
22
23 void convertToUppercase( char *sPtr )
24 {
25    while ( *sPtr != '\0' ) {
27       if ( islower( *sPtr ) ) {
28          *sPtr = toupper( *sPtr );
29       }
30
31       ++sPtr;
32    }
34 }
```

sPtr 和 *sPtr 都可更改

convertToUppercase 函式可更改
函式可更改sPtr 和 *sPtr 兩者

```
The string before conversion is: characters and $32.98
The string after conversion is: CHARACTERS AND $32.98
```

- 指向 **常數資料**的非常數指標

```
5  #include <stdio.h>
6
7  void printCharacters( const char *sPtr );
8
9  int main( void )
10 {
12    char string[] = "print characters of a string";
13
14    printf( "The string is:\n" );
15    printCharacters( string );
16    printf( "\n" );
17    return 0;
18 }

22 void printCharacters( const char *sPtr )
23 {
25    for ( ; *sPtr != '\0'; sPtr++ ) {
26       printf( "%c", *sPtr );
27    }
28 }
```

指標變數 **sPtr** 是可更改的，但是它指向的資料 ***sPtr** 則不行

sPtr 可以被 **printCharacters**函式更改

```
The string is:
print characters of a string
```

- 指向非常數資料的 **常數指標**

```
3   #include <stdio.h>
4
5   int main( void )
6   {
7      int x;
8      int y;
12     int * const ptr = &x;
13
14     *ptr = 7;
15     ptr = &y; /* error: ptr is const; cannot assign new address */
16     return 0;
17  }
```

指標 ptr 是不可更改的，但是它指向的資料 *ptr 則可以更改

```
Compiling...
FIG07_13.c
c:\examples\ch07\FIG07_13.c(15) : error C2166: l-value specifies const object
Error executing cl.exe.

FIG07_13.exe - 1 error(s), 0 warning(s)
```

課本pp. 7-16

- 指向 **常數資料**的 **常數指標**

```
3   #include <stdio.h>
4
5   int main( void )
6   {
7      int x = 5;
8      int y;
13     const int *const ptr = &x;
14
15     printf( "%d\n", *ptr );
16     *ptr = 7; /* error: *ptr is const; cannot assign new value */
17     ptr = &y; /* error: ptr is const; cannot assign new address */
18     return 0;
19  }
```

指標 **sPtr**和它指向的資料 ***sPtr**都是不可更改的

```
Compiling...
FIG07_14.c
c:\examples\ch07\FIG07_14.c(17) : error C2166: l-value specifies const object
c:\examples\ch07\FIG07_14.c(18) : error C2166: l-value specifies const object
Error executing cl.exe.

FIG07_12.exe - 2 error(s), 0 warning(s)
```

7.6 使用傳參考呼叫的氣泡排序法

- 修改氣泡排序法
  - 改成兩個函式
    - **bubbleSort( )**: 用來排序
    - **swap( )**: 用來交換元素
  - 使用**傳參考呼叫**方式
    - bubbleSort 函式中需傳遞兩個元素的位址到swap函式內

```
swap(&array[j],&array[j+1]);
```

```
1  /* Fig. 6.15: fig06_15.c  */
3  #include <stdio.h>
4  #define SIZE 10
7  int main( void )
8  {
10    int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
11    int pass;
12    int i;
13    int hold; /* temporary to swap array elements */
15    printf( "Data items in original order\n" );
18    for ( i = 0; i < SIZE; i++ ) {
19       printf( "%4d", a[ i ] );
20    }
22    /* bubble sort */
24    for ( pass = 1; pass < SIZE; pass++ ) {
27       for ( i = 0; i < SIZE - 1; i++ ) {
31          if ( a[ i ] > a[ i + 1 ] ) {
32             hold = a[ i ];
33             a[ i ] = a[ i + 1 ];
34             a[ i + 1 ] = hold;
35          }
36       }
37    }
39    printf( "\nData items in ascending order\n" );
40
42    for ( i = 0; i < SIZE; i++ ) {
43       printf( "%4d", a[ i ] );
44    }
46    printf( "\n" );
47    return 0;
48 }
```

課本pp. 6-25

```
1   /* Fig. 7.15: fig07_15.c */
4   #include <stdio.h>
5   #define SIZE 10
7   void bubbleSort( int * const array, const int size );
8
9   int main( void )
10  {
12     int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
14     int i;
15
16     printf( "Data items in original order\n" );
19     for ( i = 0; i < SIZE; i++ ) {
20        printf( "%4d", a[ i ] );
21     }
22
23     bubbleSort( a, SIZE );
24
25     printf( "\nData items in ascending order\n" );
28     for ( i = 0; i < SIZE; i++ ) {
29        printf( "%4d", a[ i ] );
30     }
32     printf( "\n" );
33     return 0;
34  }
```

陣列傳給函式時，也要傳陣列大小

圖 7.15　使用傳參考呼叫的氣泡排序法

```
37  void bubbleSort( int * const array, const int size )
38  {
39     void swap( int *element1Ptr, int *element2Ptr );
40     int pass;
41     int j;
44     for ( pass = 0; pass < size - 1; pass++ ) {
47        for ( j = 0; j < size - 1; j++ ) {
50           if ( array[ j ] > array[ j + 1 ] ) {
51              swap( &array[ j ], &array[ j + 1 ] );
52           }
53        }
54     }
55  }
56

59  void swap( int *element1Ptr, int *element2Ptr )
60  {
61     int hold = *element1Ptr;
62     *element1Ptr = *element2Ptr;
63     *element2Ptr = hold;
64  }
```

Swap函式的原型在此宣告，因此只有bubbleSort函式才能呼叫swap( )

```
Data items in original order
   2   6   4   8  10  12  89  68  45  37
Data items in ascending order
   2   4   6   8  10  12  37  45  68  89
```

7.7 sizeof運算子

- 用來計算出任何資料的大小(bytes)
  - 在程式編譯(compiler)時計算的
  - 如： **float arr1[20];**
    printf("%d", sizeof(arr1));

    arr1 共占用 20*4 = 80 個bytes
- 也可計算出陣列中有多少個元素
  - 如： **int num = sizeof(arr1) / sizeof(arr1[0])**

增進效能的小技巧 7.2

sizeof 是一個編譯時期的運算子，所以不會增加執行時期的負擔

課本pp. 7-21

```
1  /* Fig. 7.16: fig07_16.c  */
4  #include <stdio.h>
6  size_t getSize( float *ptr );
7
8  int main( void )
9  {
10    float array[ 20 ];
12    printf( "The number of bytes in the array is %d"
13            "\nThe number of bytes returned by getSize is %d\n",
14            sizeof( array ), getSize( array ) );
15    return 0;
16 }
17
19 size_t getSize( float *ptr )
20 {
21    return sizeof( ptr );
22 }
```

型別 size_t 也就是sizeof傳回值的型別

```
The number of bytes in the array is 80
The number of bytes returned by getSize is 4
```

可攜性的小技巧 7.2

用來存放某一種資料型別的位元組個數，可能會隨著系統的不同而有所差異。當你撰寫的程式與資料型別的大小有關，而且必須在數種電腦上執行時，你最好使用 `sizeof` 來判斷資料型別所佔用的位元組個數。

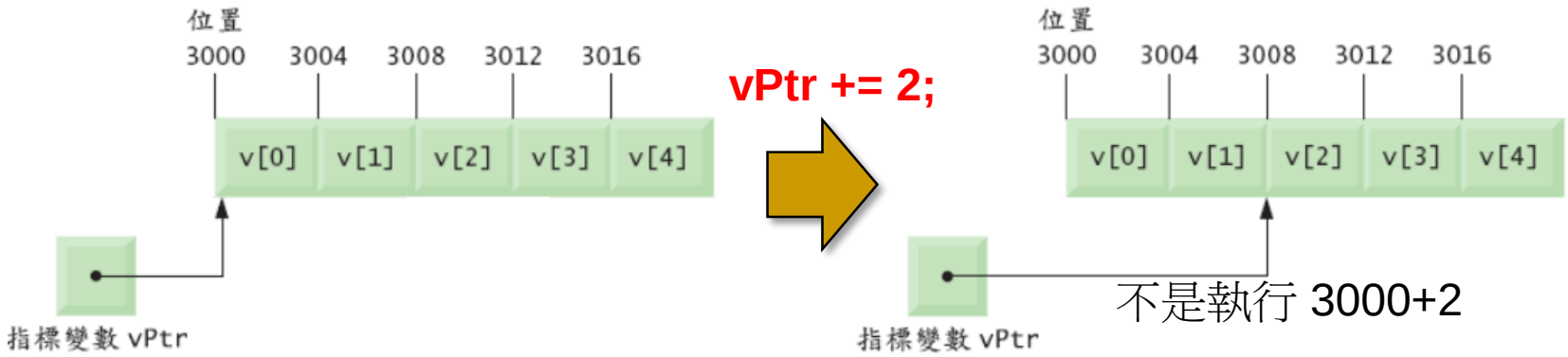
```
sizeof(char)   = 1
sizeof(short) = 2
sizeof(int) = 4
sizeof(long) = 4
sizeof(float) = 4
sizeof(double) = 8
sizeof(long double) = 8
```

課本pp. 7-23

7.8 指標運算式和指標的算術運算

- 指標可作為運算元的運算子
  - 指標可進行遞增(++)、遞減(--)、加上(+, +=)或減去(-, -=)等運算
  - 範例：

```
int v[5];
int *vPtr;
vPtr = v; //或寫成 vPtr = &v[0];
```



Q: vPtr++; 指標會移到哪個位置呢?

- 指標變數可互相進行相減
  如：`x = v2Ptr – vPtr;`

  此敘述句會將介於vPtr與vPtr2之間的陣列元素個數指定給x，因此兩個指標必須指向同一個陣列才行

常見的程式設計錯誤 7.5

對一個不是指向陣列的指標進行指標的算術運算

常見的程式設計錯誤 7.6

對兩個不是指向同一陣列的指標進行相減或比較

常見的程式設計錯誤 7.7

在使用指標的算術運算時，指標超出了陣列的範圍

課本pp. 7-25

練習

- 請問下列程式的執行結果

```
#include <stdio.h>
int main(void)
{
    int b[ ]={10, 20, 30, 40, 50, 60, 70};
    int *bptr=b+3;
    int i;
    for(i=0; i<4; i++)
        printf("*(bptr+%d) = %d \n", i, *(bptr+i));
    for(i=-1; i>-4; i--)
        printf("*(bptr+%d) = %d \n", i, *(bptr+i));
    return 0;
}
```

練習

- 請問下列程式做些什麼事?

```
#include <stdio.h>
int mystery2(const char *s);
int main(void){
      char string[80];
      printf(“Enter a string: ”);
      scanf(“%s”, string);
      printf(“%d\n”, mystery2(string));
      return 0;
}
int mystery2(const char *s){
      int x;
      for( x=0; *s !=‘\0’; s++)
          x++;
      return x;
}
```

課本pp. 7-54, EX. 7.20題

練習

- 使用下列函式來計算某個陣列中所有值的總和，其中陣列個數及元素內容自行設定

  int sum(int *bptr, int num)

- 使用下列函式來取得某個陣列的最小元素值

  int min(int *bptr, int num)

- 使用下列函式來交換兩個數值

  void swapTwo(int *a, int *b)

練習

- 寫一個程式，利用隨機函式產生6個1~49的整數存入陣列中，並寫一個函式來取得陣列中的最大值、最小值及平均值。
  - 函式名稱為
    void findMaxMinMean(int *aPtr, int *maxPtr, int *minPtr, float *meanPtr)

```
27 35 47 41 2 31
max: 47
min: 2
mean: 30.50
```