【第七講】

指標

講師: 李根逸 (Ken-Yi Lee), E-mail: feis.tw@gmail.com

★ ★ ★ ★

★ ★ ★ ★

課程大綱

沒有『指標』的世界

- **『指標』是一種資料型態，用來儲存『記憶體位址』**
  - 一般情況下，我們是不需要『指標』這種東西的。
- **但是 C 語言中為了解決某些問題或提昇程式效率，在某些情況下需要使用『指標』。**
- **先來看看有什麼問題是以前不能處理的：**
  - 在被呼叫的函式中修改非該函式內的變數內容
  - 直接複製陣列
    - 例如呼叫函式時要將陣列作為函式引數複製到函式內
  - 直接複製字串
    - 例如呼叫函式時要將字串作為函式引數複製到函式內
  - 動態配置記憶體
    - 例如要動態改變陣列的大小

指標使用的情境 [1]

- 在被呼叫的函式中修改非該函式內的變數內容
  - 當我們將變數送入函式執行時，是使用『傳值呼叫』的方式。意味著變數的值會被當做引數值而複製一份進函式內部做為參數。在函式內部對參數做任何的變動不會改變到原本的引數值：

```
void addone(int n) {
  n = n+1;
}

int main() {
  int a = 3;
  addone(a);              /* 複製 a 的值給 addone */
  printf("%d", a);
  return 0;
}
```

對 addone 來說他只是得到一個整數的複製品，無法知道整數原本存放的地方或來源

指標使用的情境 [2]

- 直接複製陣列：

```
int v[5] = {1, 2, 3, 4, 5};
int n[5];
n = v;              // 語法錯誤
```

  - 例如在呼叫函式時要將陣列作為函式引數複製到函式內

```
:
void print(int n[3]) {
  for (int i = 0; i < 3; ++i) {
    printf("%d ", n[i]);
  }
}

int main() {
  int v[3] = {1, 2, 3};
  print(v);
  printf("\n");
  return 0;
}
```

指標使用的情境 [3]

- 直接複製字串：

```
char v[6] = "Hello";
char n[6];
n = v;              // 語法錯誤
```

  - 例如在呼叫函式時要將字串作為函式引數複製到函式內

```
:
void print(char n[6]) {
  for (int i = 0; i < 5; ++i) {
    printf("%c", n[i]);
  }
}

int main() {
  char v[6] = "Hello";
  print(v);
  printf("\n");
  return 0;
}
```

C 風格字串是利用陣列的方式儲存

指標使用的情境 [4]

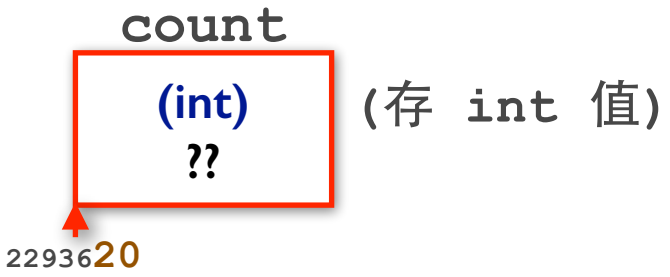
- 動態配置記憶體
  - 動態改變陣列大小

```
int v[5];
int n[10];
v = n;              // 語法錯誤
```

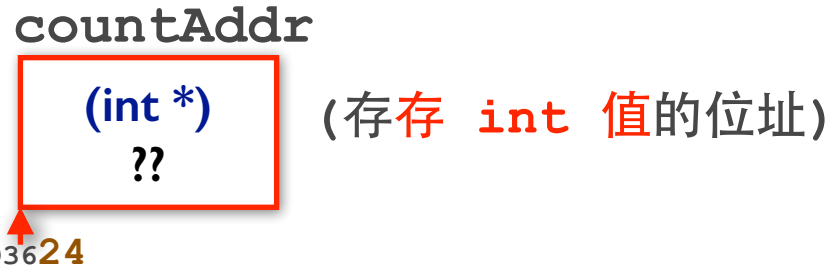
- 我們在儲存資料前需要先配置記憶體，如果在一開始未知資料的總數時，我們要怎麼配置？
  - 例如要寫一個程式，讓使用者輸入任意筆資料後儲存起來要怎麼辦？

指標變數宣告 (type *)

- 指標 (Pointer) 是 C/C++ 語言的一大特色，是一種儲存記憶體位址的資料型態
- 指標變數宣告語法：
  - 資料型態 *變數名稱;
    - 表示變數名稱內存放的是一存放該資料型態值的記憶體位址
- 宣告指標變數與其他變數的差別：
  - **int count;**



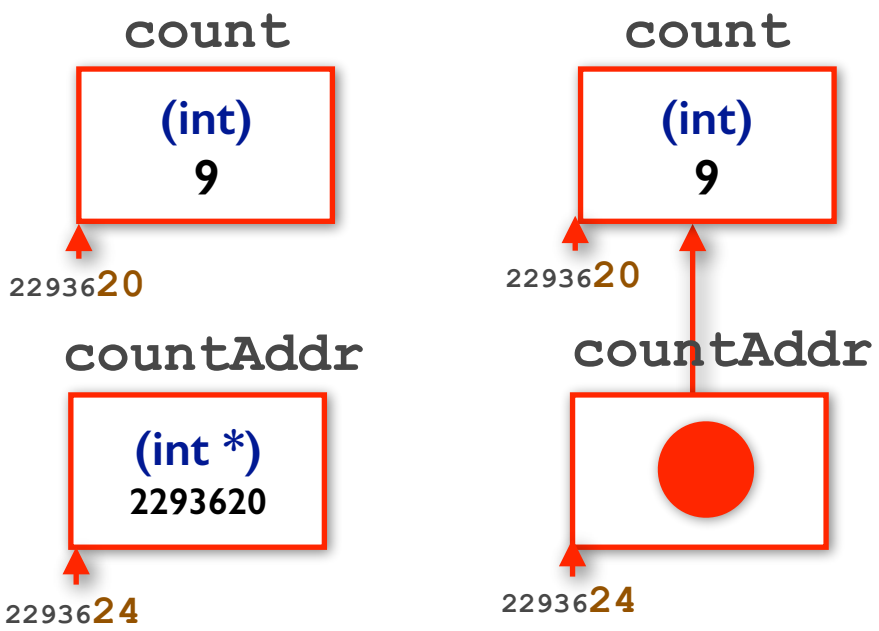
  - **int *countAddr;**



取址運算子 (&)

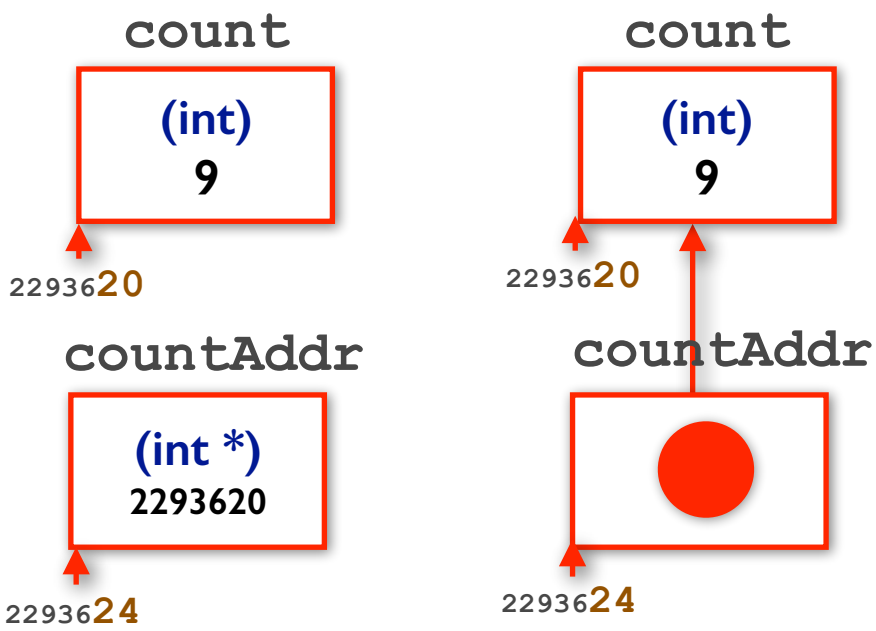
- 變數宣告後依照資料型態會佔據一定的記憶體空間，並具有一個在記憶體的位址。我們可以利用取址運算子 (&) 去取得變數的記憶體位址：
  - **int count** = 9;
  - **int *countAddr = &count**;

表示式	資料型態	值
count	int	9
&count	int *	2293620
countAddr	int *	2293620



間接運算子 (*)

- 相對地，我們可以利用間接運算子 (*) 從位址取得位於該位址的變數 (別跟宣告指標用的*搞混)
  - **int count** = 9;
  - **int *countAddr** = **&count**;
  - **int result** = ***countAddr**;



表示式	資料型態	值
count	int	9
&count	int *	2293620
countAddr	int *	2293620
*countAddr	int	9
result	int	9

注意這裡出現的兩個星號 (*) 不同！

【範例】pointer.cpp

函式傳值呼叫

- 當我們將變數送入函式執行時，是使用『傳值呼叫』的方式。意味著變數的值會被當做引數值而複製一份進函式內部做為參數。在函式內部對參數做任何的變動不會改變到原本的引數值：

```
void addone(int n) {
  n = n+1;
  return;
}

int main() {
  int a = 3;
  addone(a);              /* 複製 a 的值給 addone */
  printf("%d", a);
  return 0;
}
```

對 addone 來說他只是得到一個整數，無法知道整數原本存放的地方或來源

函式傳址呼叫

- 我們可以將變數的『記憶體位址』作為引數值複製進入函式執行。此時在函式內部對參數用『間接運算子』指定新的值時就會改變原本的變數值。
  - 其實這也是函式傳值呼叫的一種，只是該值是個位址

```
void addone(int *n) {
  *n = *n+1;
  return;
}

int main() {
  int a = 3;
  addone(&a);          /* 複製 a 所在位址給 addone */
  printf("%d", a);
  return 0;
}
```

對 addone 來說他得到了一個位址，經由間接運算子 (*) 可以取得並指定該位址上的值

可以看成 C 語言只能用複製傳值的方式呼叫函式，而位址也是一種值

【範例】交換與排序

- 試寫一函式 **void swap(int *, int *)**，將輸入的兩個整數參數的值交換
  - 例如:
    - a = 1，b = 2，執行完 **swap** 後，a = 2, b = 1

【範例】swap.cpp

- 試寫一函式 **void sort(int *, int *)**，將輸入的兩個整數參數的值由小到大排
  - 例如:
    - a = 1，b = 2，執行完 **sort** 後，a = 1, b = 2
    - a = 2，b = 1，執行完 **sort** 後，a = 1, b = 2

【範例】sort.cpp

傳值還是傳址？

- 簡言之，在 C 語言裡如果你想要讓其他函式可以幫你修改變數的值時就需要傳該變數的位址
  - 傳值的設計讓不同函式之間的非全域變數是完全沒有關係的、是不會互相影響的，可以避免污染與干涉！
  - 要藉由『呼叫函式』來修改目前所在函式的變數值有兩個方式：
    - 接收回傳值：

```
var = func();
```

    - 傳送變數的位址 ：

```
func(&var);
```

傳陣列只能傳址

- 能傳值就傳值，可以避免函式之間的間接汙染！
  - 變數位址就好像變數真實的名字一樣，得到的函式可以為所欲為！

【範例】讀出與印出

- 試寫兩函式，其中 **read** 函式可讀入一成績，並用 **show** 函式將成績印出

接收回傳值

```
#include <stdio.h>
#include <stdlib.h>
int read();
void show(int);

int main() {
  int grade = read();
  show(grade);
  system("pause");
  return 0;
}
```

傳送變數位址

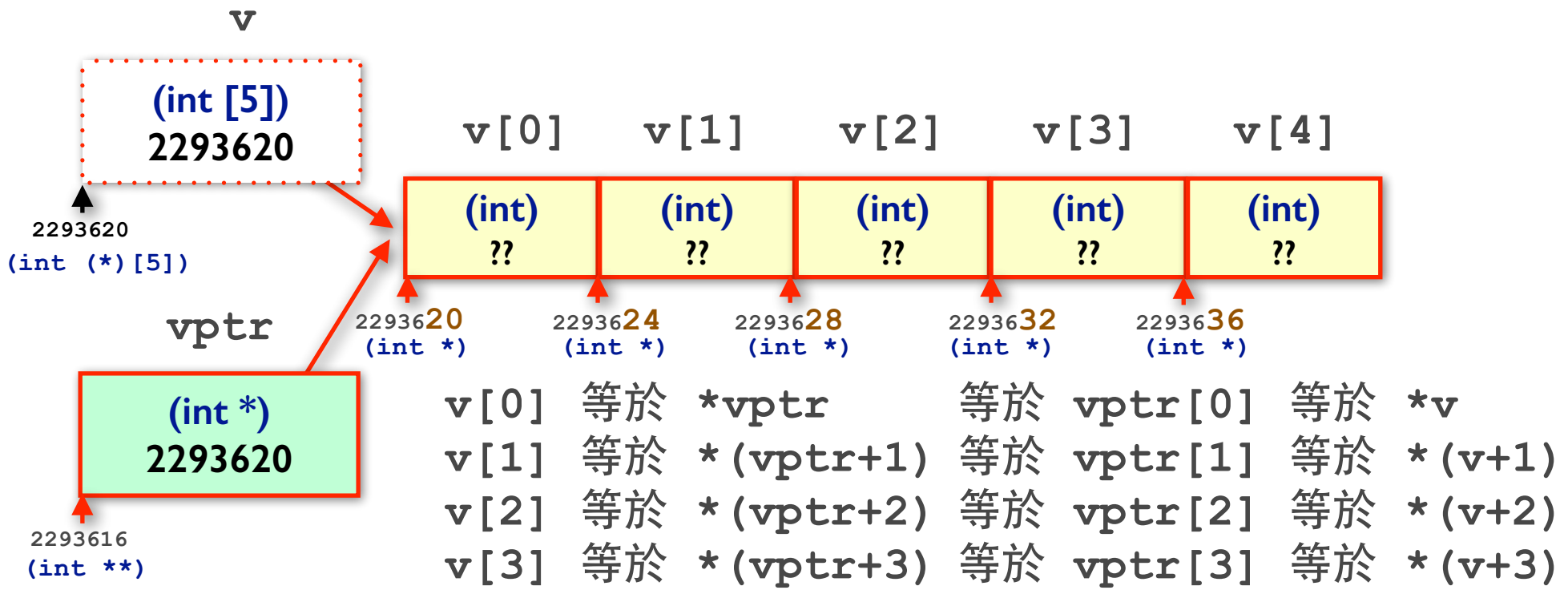
```
#include <stdio.h>
#include <stdlib.h>
void read(int *);
void show(int);

int main() {
  int grade;
  read(&grade);
  show(grade);
  system("pause");
  return 0;
}
```

【範例】read_show.cpp

指標與陣列 [1]

- 指標與陣列關係相當密切，似乎是一體的兩面，但是又有著蠻多的不同：
  - **int v[5];** /* 會配置五個(**int**)的記憶體空間 */
  - **int *vptr = v;** /* 會配置一個(**int** *)的記憶體空間 */



指標與陣列 [2]

- **指標與陣列的關係：**
  - 陣列型態可以轉型為指向該陣列第一個元素的指標
  - 指標可以像陣列一樣使用 [] 運算子來存取記憶體，此時會以該指標指向的變數做為陣列的第一個元素，以該指標指向的變數資料型態做為陣列元素型態
- **指向陣列元素的指標可以對整數做 + 或 - 運算：**
  - 加 N：指標會指向往後走 N 個元素
  - 減 N：指標會指向往前走 N 個元素
- **對兩個指向同陣列元素的指標做 - 的算術運算：**
  - 得到兩個指標分別指向的元素差了幾號

【範例】比大小

- 寫一個可以讓使用者輸入五個整數，回傳最大值的函式 **max1v:**
  - `int max1v( int * );`

可以用指標變數來接收陣列的位址值

- 寫一個可以讓使用者輸入任意個整數，回傳最大值的函式 **max2v:**
  - `int max2v( int *, int );`

除了要傳入起始位址，還要傳入元素個數

【範例】max.cpp

【範例】用指標對陣列存取 [1]

- 將下述程式迴圈的部份用指標改寫：

```
int main() {
  int v[10];
  for (int i = 0; i < 10; i++) {
      v[i] = 0;
  }
  return 0;
}
```

```
int main() {
  int v[10];
  int *p = v;
  for (int i = 0; i < 10; i++) {
      *p = 0;
      p++;
  }
  return 0;
}
```

【範例】用指標對陣列存取 [2]

- 將下述程式迴圈的部份用指標改寫：

```
int main() {
  int v[10];
  for (int i = 0; i < 10; i++) {
      v[i] = 0;
  }
  return 0;
}
```

```
int main() {
  int v[10];
  for (int *p = v; p != &v[10]; p++) {
      *p = 0;
  }
  return 0;
}
```

有比較好嗎？