C++程式設計基礎
week 5

陳毅

Zerojudge

- b515: 摩斯電碼-商競103
  - https://zerojudge.tw/ShowProblem?problemid=b515

本週概要

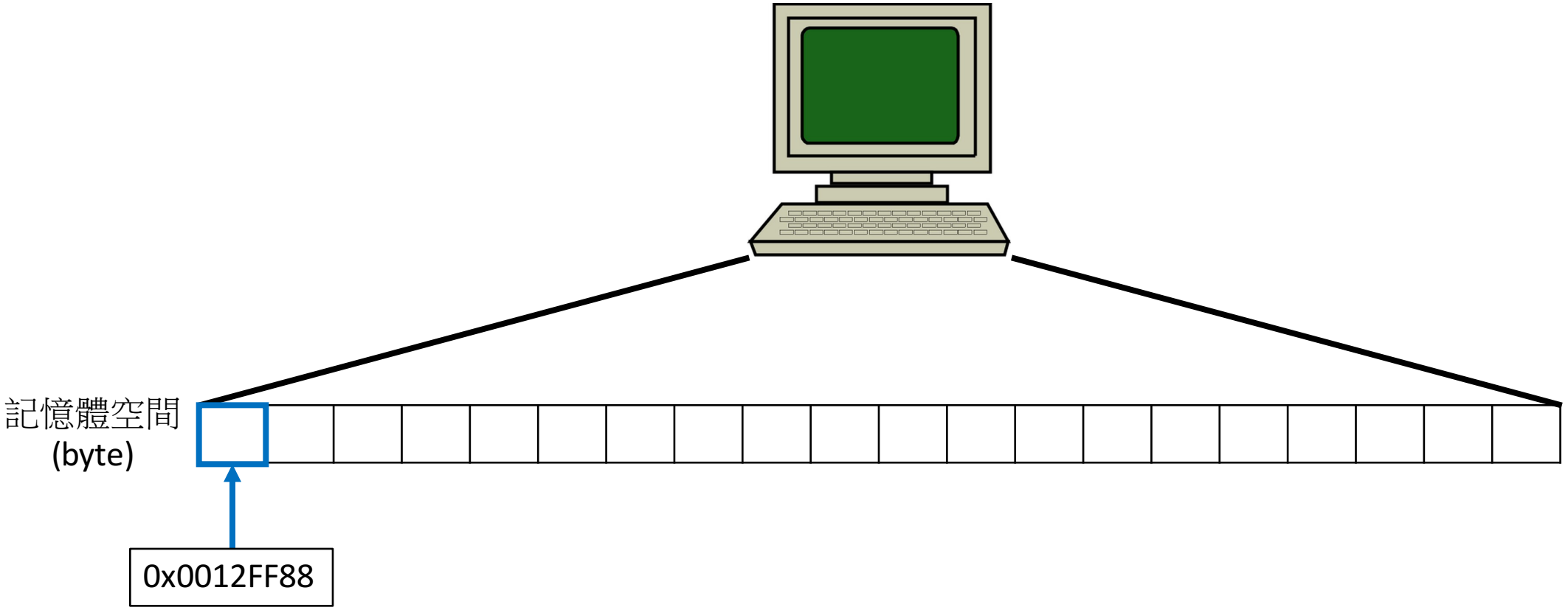
- 指標
- 「製作OOXX遊戲」講解

指標

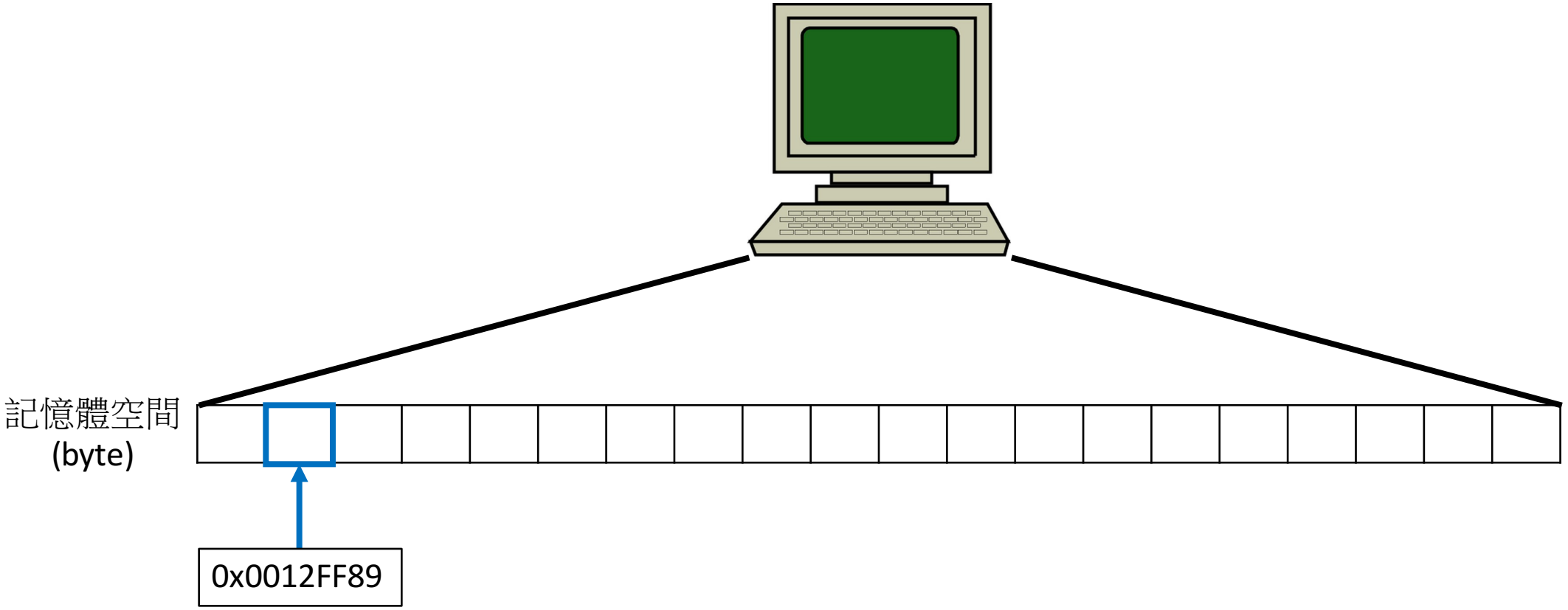
今天過後，你要學會...

- 宣告一個用來儲存「**記憶體位址**」的變數，此變數稱作**指標**。
- 使用「位址運算符 &」得到一個變數的**記憶體位址**。
- 使用「間接運算符 *」得到一個記憶體位址所儲存的**值**。
- 陣列的指標
  - **指標運算**
  - **多重指標**
- 使用**指標**作為函數的引數（參數），來傳遞資料。

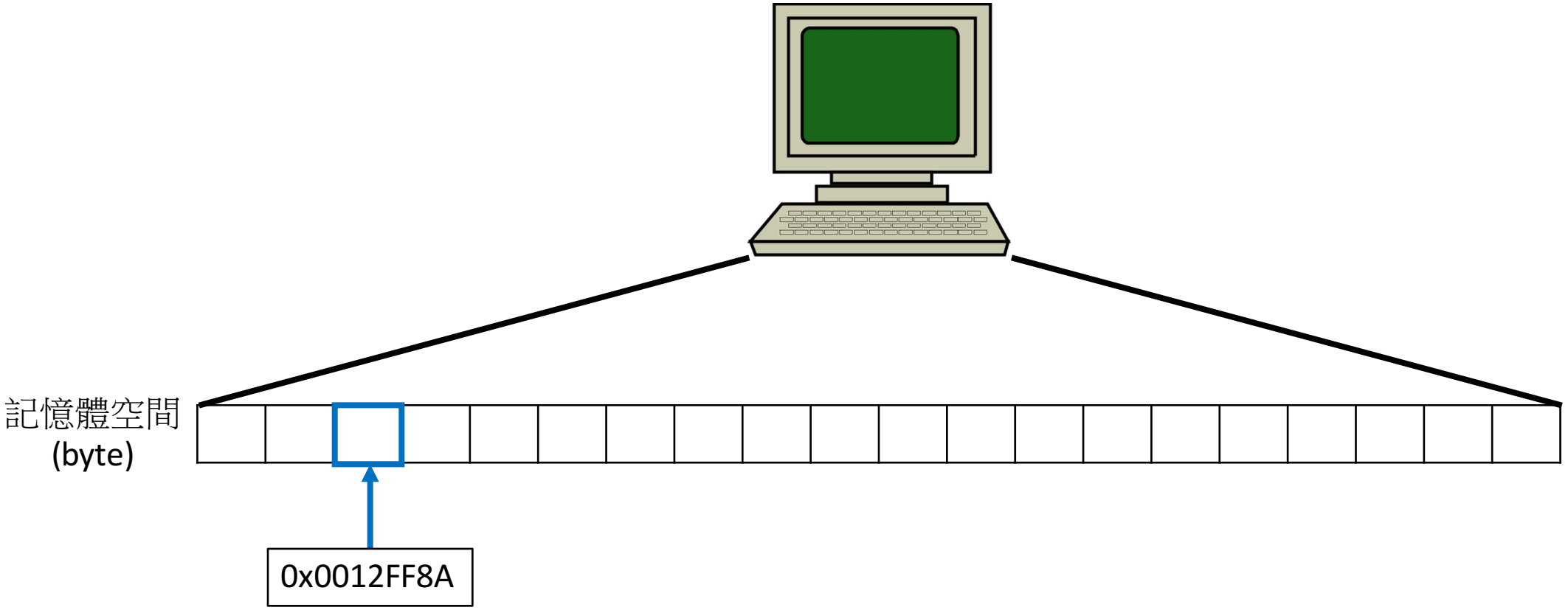
電腦的記憶體



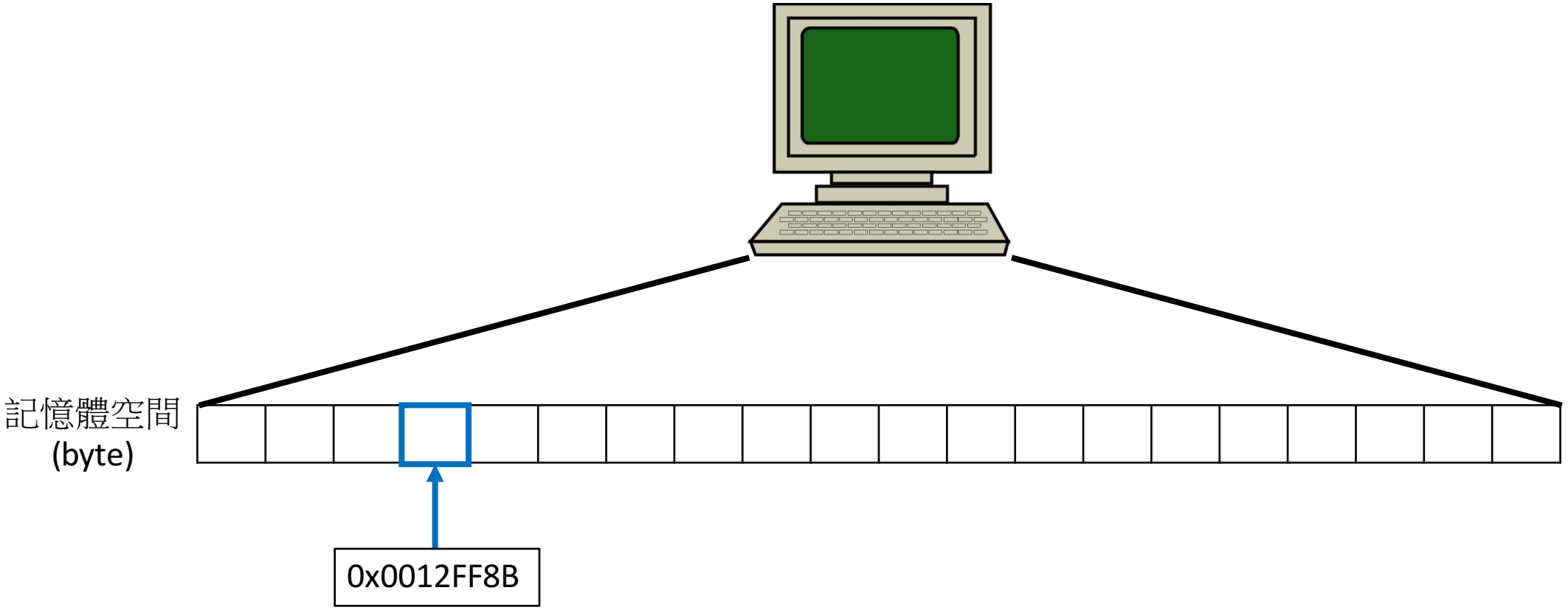
電腦的記憶體



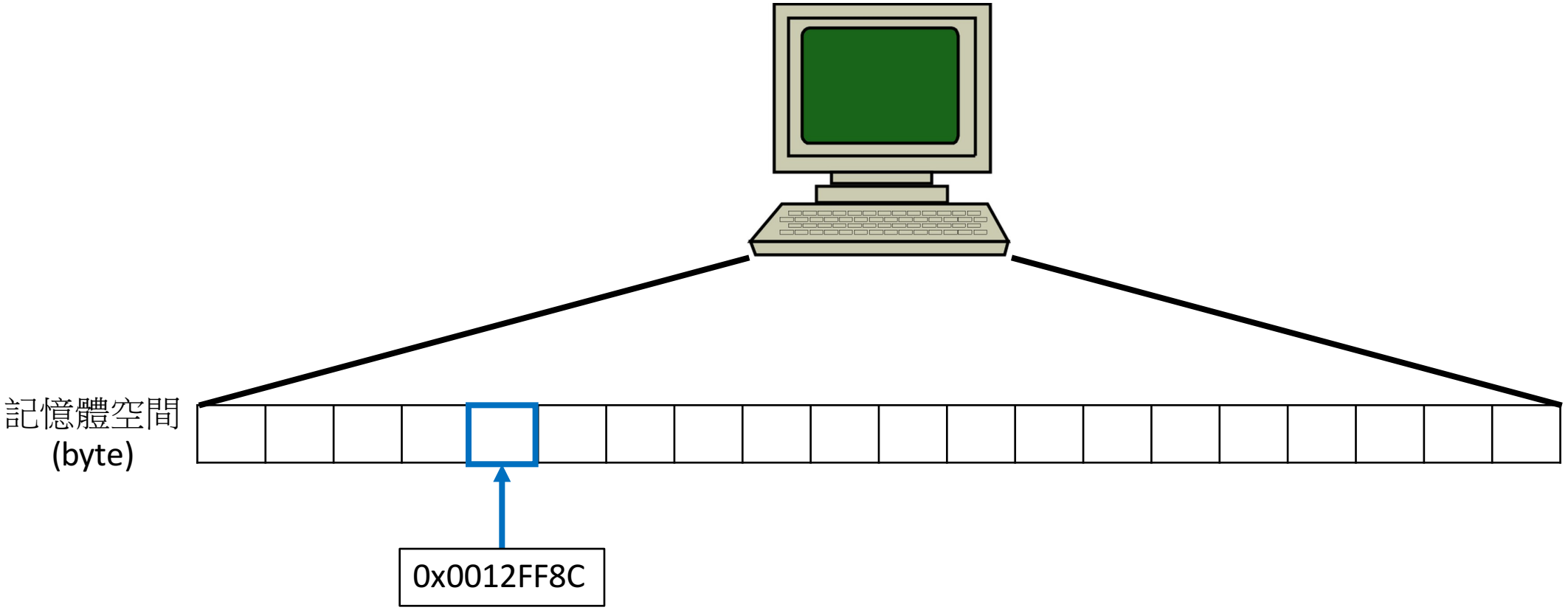
電腦的記憶體



電腦的記憶體



電腦的記憶體



複習：sizeof 運算子 (week 4)

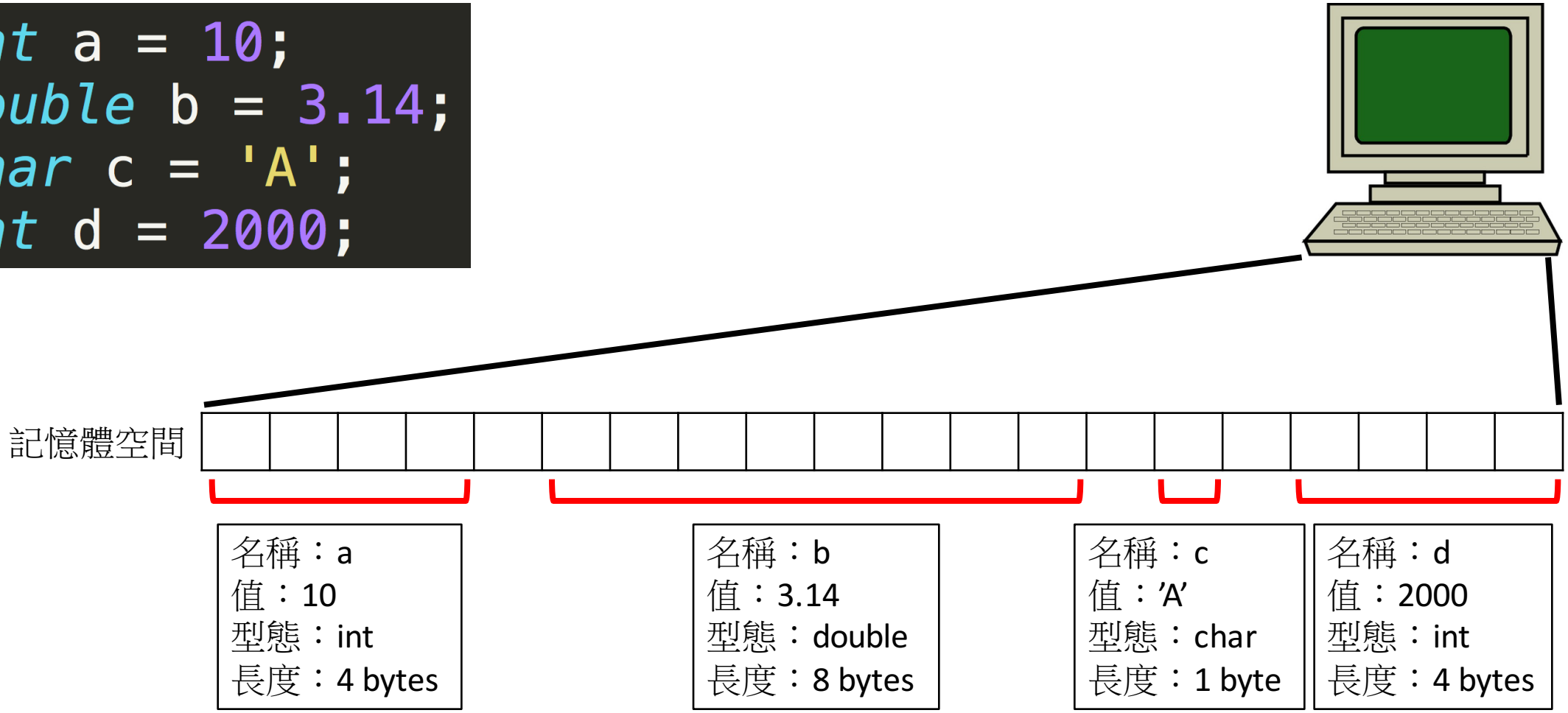
- 用來計算程式中**變數所佔用的記憶體大小**，這個運算子在動態配置記憶體時有很大的用處。

```
int a;
char b;
float c;
double d;
bool e;
cout << "int: " << sizeof(a) << endl;
cout << "char: " << sizeof(b) << endl;
cout << "float: " << sizeof(c) << endl;
cout << "double: " << sizeof(d) << endl;
cout << "bool: " << sizeof(e) << endl;
```

4-2.cpp

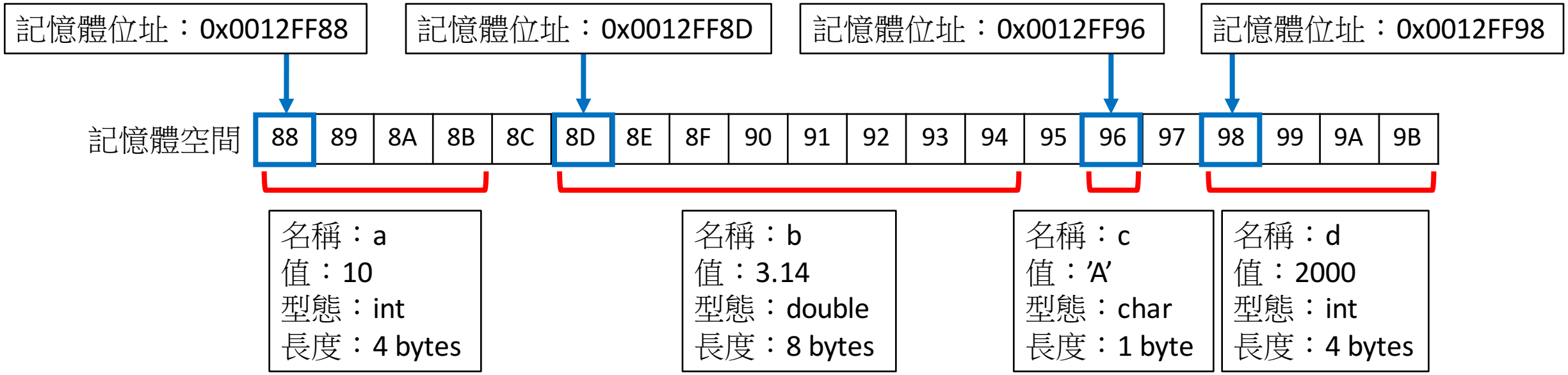
關於電腦如何儲存一個變數的資料

```
int a = 10;
double b = 3.14;
char c = 'A';
int d = 2000;
```



變數的記憶體位址

- 儲存一個變數的第一個記憶體位址，即為該變數的記憶體位址



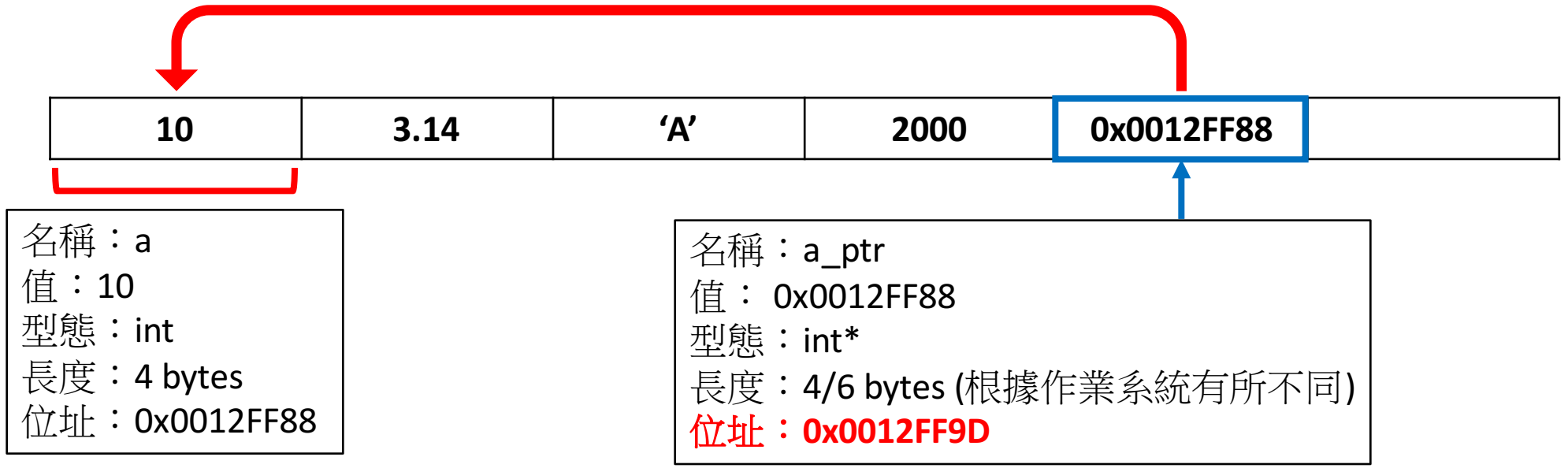
知道變數的記憶體位址能做什麼？

- 存取變數的值
  - 可以透過記憶體位址，去存取變數的值。
- 方便傳遞
  - 將一個元素很多的陣列傳遞給函數，要將每個元素都複製一遍傳遞過去。但若知道陣列的記憶體位址，那只要傳遞記憶體位址，函數就可以知道陣列在哪。
- 可用來構築資料結構
  - 如：堆疊(stack)、佇列(queue)、鏈結串列(linked list)、二元樹(binary tree)。

指標（Pointer）

- 指標是一種資料型態，用來儲存記憶體位址。
- 指標本身也具有記憶體空間。

```
int *a_ptr = &a;
```



宣告指標

```
資料型態 *指標變數;
```

- 宣告指標變數與宣告一般變數的方法類似，只是在指標變數前面加上「*」或是在資料型態後面加上「*」。

容易在同時宣告多個變數時出現誤用

誤用

- 宣告兩個指標變數ptr1與ptr2。

```
int* ptr1, ptr2;
```

ptr1為指標變數
ptr2為int變數

```
int *ptr1, *ptr2;
```

正確

指標與變數的參考

- 取得變數的**記憶體位址**
  - 「&」稱作**位址運算符(address-of operator)**，是用來取得變數的位址，也稱作**參考運算符號(reference operator)**。

```
&變數名稱
```

- 取得一個記憶體位址所儲存的**值**
  - 「*」稱作**間接運算符號(indirect operator)**，是用來取得參考位址內的值，也稱作**反參考運算符號(de-reference operator)**。

```
*指標名稱
```

5-1.cpp, 5-2.cpp

注意事項

- 指標需指向正確的型態

```
float number;
int *ptr = &number;
```

- 不可用&運算子對常數或運算式取值

```
int *ptr = &3;
int *ptr2 = &(num1+num2);
```

- 不可在指標並未指向任何記憶體時，使用*運算子

```
int *ptr;
cout << *ptr;
```

練習

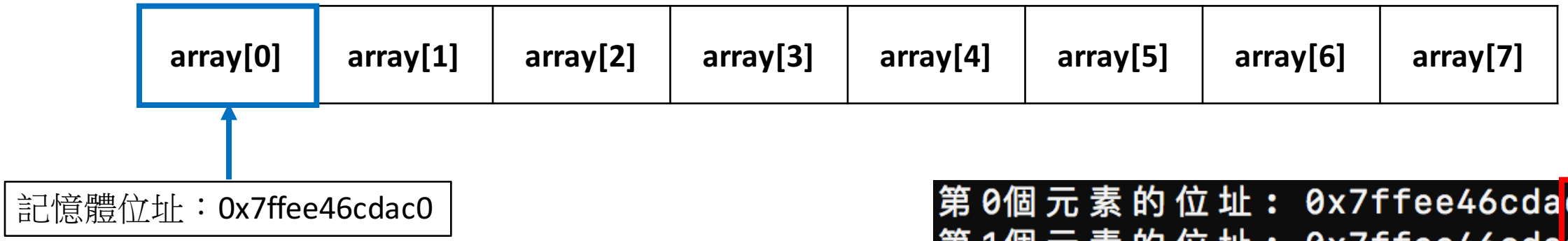
- 課本範例程式：8-04

- 如果執行這份程式碼，會看到什麼結果？ (5-3.cpp)

```
int array[8];
for (int i = 0 ; i < 8 ; i++){
    cout << "第" << i << "個元素的位址: " << &array[i] << endl;
}
```

陣列的指標

- 一維陣列（以int[8]為例）



array[i] = array[0] + i * sizeof(int)

一個int佔用4個bytes

```
第 0個 元 素 的 位 址 :  0x7ffee46cdac0
第 1個 元 素 的 位 址 :  0x7ffee46cdac4
第 2個 元 素 的 位 址 :  0x7ffee46cdac8
第 3個 元 素 的 位 址 :  0x7ffee46cdacc
第 4個 元 素 的 位 址 :  0x7ffee46cdad0
第 5個 元 素 的 位 址 :  0x7ffee46cdad4
第 6個 元 素 的 位 址 :  0x7ffee46cdad8
第 7個 元 素 的 位 址 :  0x7ffee46cdadc
```

陣列的指標

- 當一個陣列被宣告時，它的名稱可當作指標使用，指向該型態陣列的第一個元素。

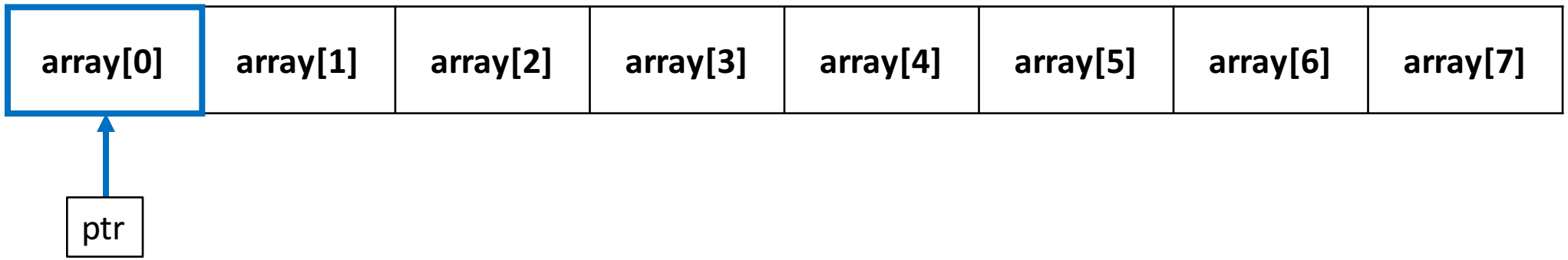
```
array == &array[0]
```

指標運算

- 指標型態不像一般變數有四則運算，指標只有加減法可用。
  - +
  - -
  - ++
  - --
- 對於指標變數使用加減法的意義：**將指標往前（後）移動n個該型態的記憶體**。

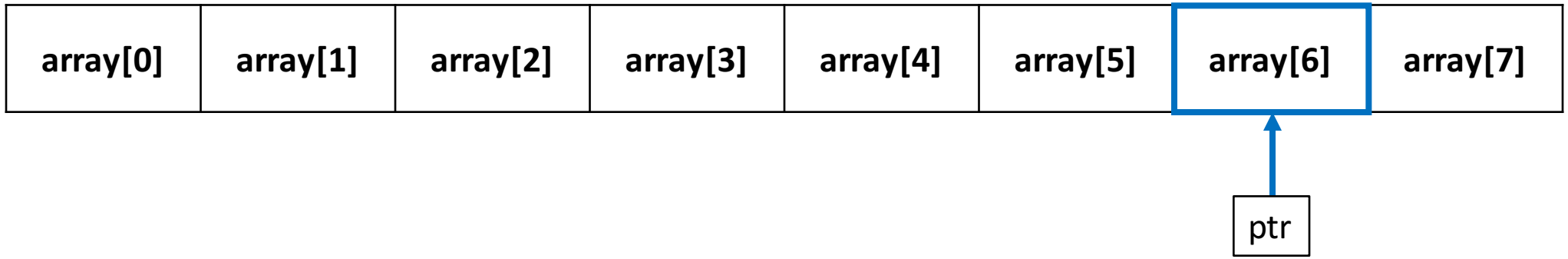
5-4.cpp

指標運算



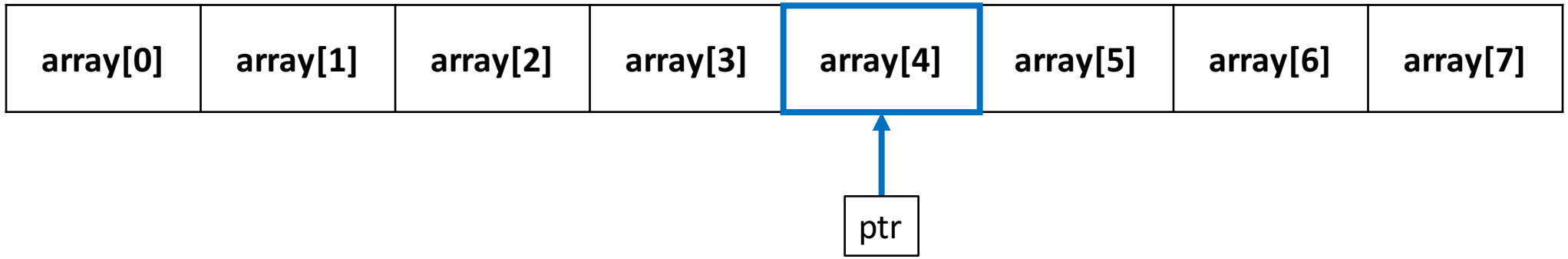
- **int *ptr;**
- **ptr = array;**
- ptr = ptr + 6;
- ptr = ptr - 2;
- ptr = ptr - 3;

指標運算



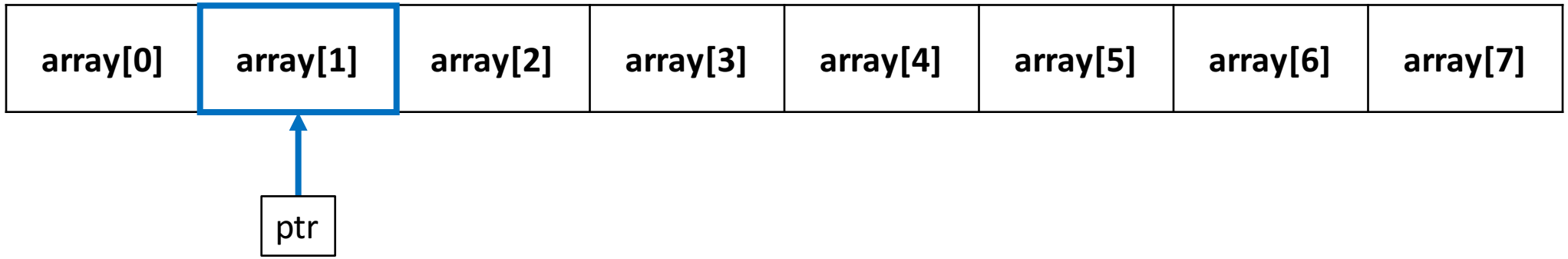
- int *ptr;
- ptr = array;
- **ptr = ptr + 6;**
- ptr = ptr - 2;
- ptr = ptr - 3;

指標運算



- int *ptr;
- ptr = array;
- ptr = ptr + 6;
- **ptr = ptr - 2;**
- ptr = ptr - 3;

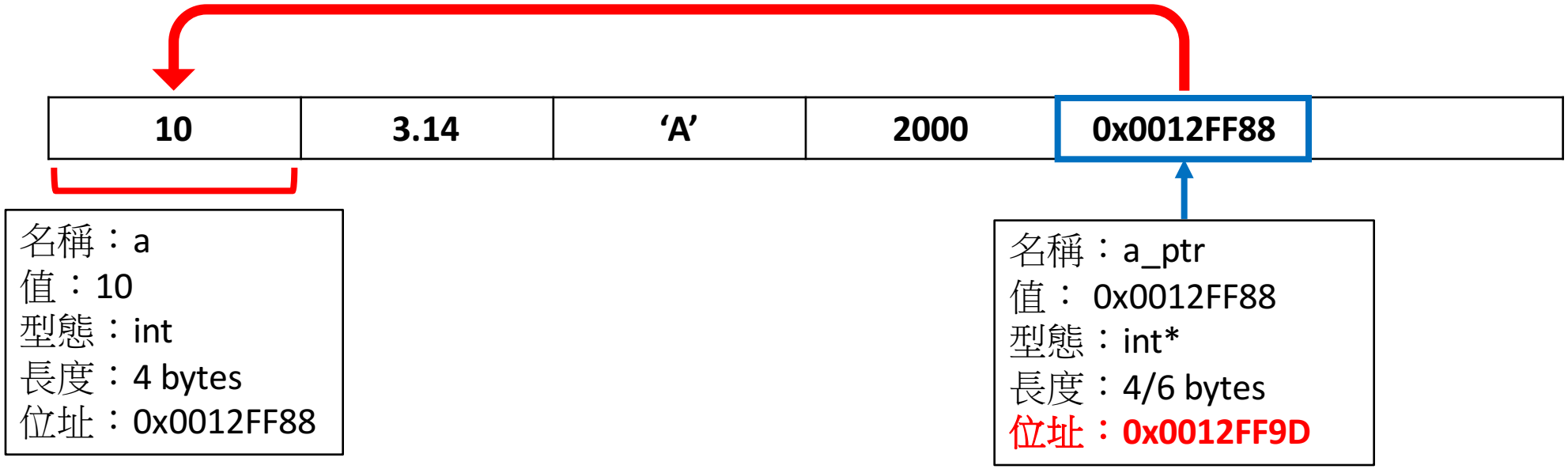
指標運算



- int *ptr;
- ptr = array;
- ptr = ptr + 6;
- ptr = ptr - 2;
- **ptr = ptr - 3;**

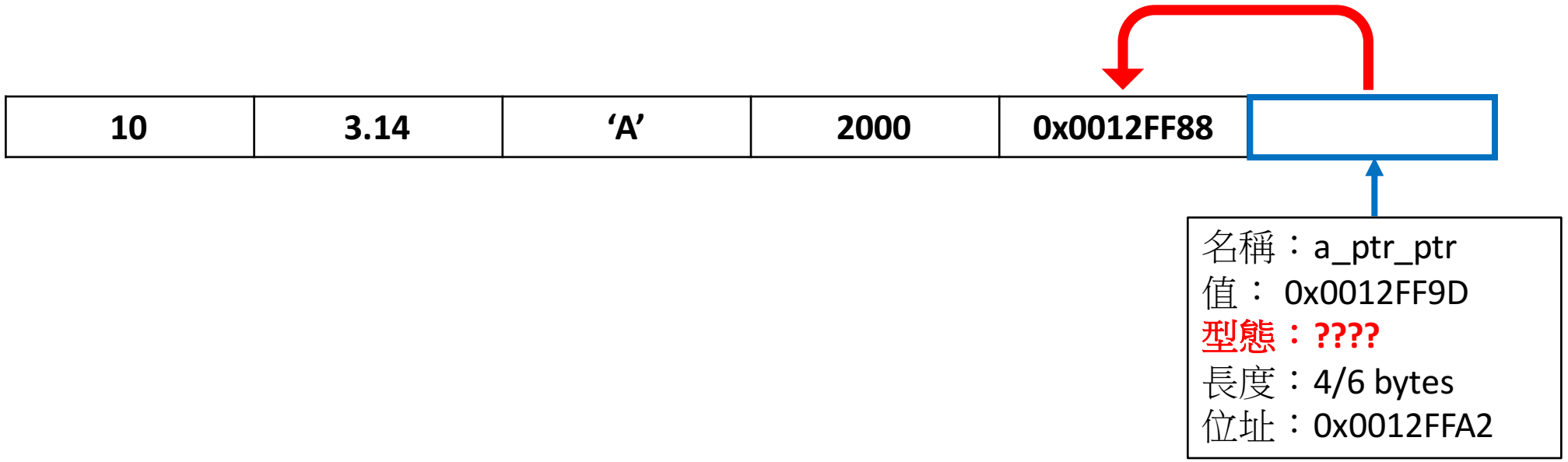
多重指標

- 由於指標本身也具有記憶體位址，所以宣告一個**指向指標型態變數的指標**是合法的。



多重指標

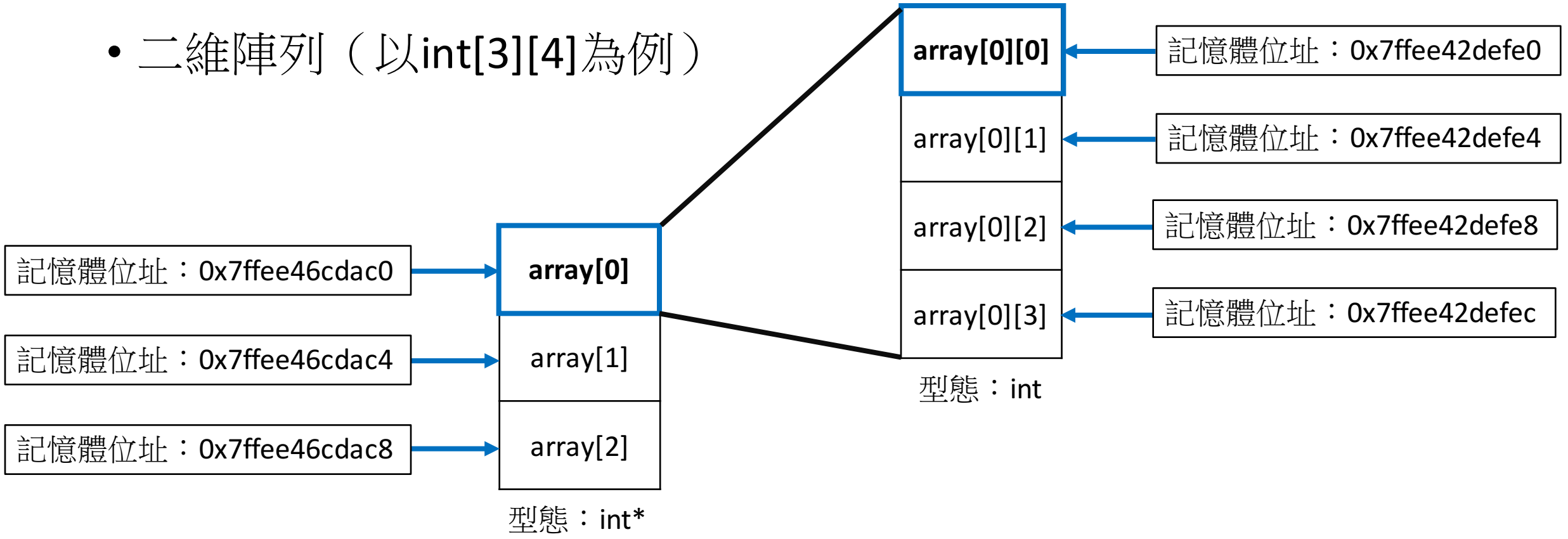
- 變數型態：int <------> 指標型態：int*
- 變數型態：int* <------> 指標型態：int**
- 變數型態：int** <------> 指標型態：int***



5-5.cpp

陣列的指標

- 二維陣列（以int[3][4]為例）



array[i] = array[0] + i * sizeof(int*)

一個int*佔用4/6個bytes

array[i][j] = array[i][0] + j * sizeof(int)

一個int佔用4個bytes

將指標傳遞進函數

- 在定義函數雛型時，可將引數型態設定為**指標型態**。
- 在傳遞大陣列時，傳遞陣列指標比傳遞整個陣列來得快許多。
- 此種方式稱作「call-by-reference」。

函數雛型(Declaration)

用來告訴編譯器，這個程式會有哪些函數。

```
型態 函數名稱(引數1型態, 引數2型態, ...);
```

函數宣告(Definition)

用來定義一個函數實際的執行內容。

```
型態 函數名稱(引數1, 引數2, ...){
    程式敘述;
    ...
}
```

複習：引數的傳遞 (week 4)

- 每一個函數都是**獨立**的，一般來說，函數只了解自己程式區塊的資料，並不認識函數外的任何變數。
- 當函數要修改到外部資料時，就必須將資料以引數的方法傳遞進函數。
- 傳遞方法分兩種：Call-by-Value、Call-by-Reference。
  - Call-by-Value：將所傳的變數**資料**複製一份，儲存在函數所宣告的自訂變數中。
  - Call-by-Reference：將所傳的變數之**記憶體位址**傳遞進函數，可以直接修改變數資料。

練習

- 變數交換程式
  - 撰寫一個程式，具有一個自訂函數swap()，可將傳入之兩個變數的值交換。
- 字串的比較
  - 撰寫一個程式，具有一個自訂函數Mystrcmp()，利用指標傳入兩個字串（字元陣列），並比較是否完全相同，如果相同回傳0，如果不同則回傳-1。

挑戰 (week 4)

- 製作OOXX遊戲
  - 遊戲規則
    - 兩個玩家，在3x3的方格中，依序填入O和X，先連成一條線的人獲勝。
  - 程式設計思路
    - 需要一個變數來儲存現在的盤面。（hint: 可使用二維陣列）
    - 需要處理玩家的輸入，例如：要將O和X填入哪個格子中。（hint: 格式化輸入）
    - 需要更新遊戲盤面，並判斷玩家的輸入是否正確（hint: 函數）
    - 需要讓玩家看到遊戲盤面。（hint: 函數、格式化輸出）
    - 玩家每動一步，就要更新一次盤面，並判斷遊戲是否結束了。（hint: 函數）
  - 進階挑戰
    - 將遊戲擴增為「五子棋」
    - 更改遊戲規則，製作「踩地雷遊戲」

下週預計課程內容

- 指標
  - **動態記憶體**
    - **配置與釋放**
    - **動態陣列**
- 挑戰：威康生命遊戲

練習

- HackerRank (Practice -> C++ -> Introduction)
  - Pointer
- 作業

作業

- 從以下題目任選兩題完成，下次上課時找助教檢查。
  - a417：螺旋矩陣
  - a693：吞食天地
  - a694：吞食天地二
  - b836：kevin戀愛攻略系列題-2 說好的霸王花呢??
  - c315：I, ROBOT 前傳
  - c381：聖經密碼
  - d115：數字包牌
- Reading: 課本Ch8.1~Ch8.3.4
- 若遇到作業問題，歡迎隨時寄信至：r07922059@ntu.edu.tw