程式語言一　講義

陣列

Hanly《C 語言詳論》6e

§ 1　名詞速查表

中文	English	一句話定義	科目	章節
陣列	array	一排連續、同型態的變數，共用一個名字	程式語言一	陣列
索引	index	存取元素的編號，從 0 開始；`arr[i]`	程式語言一	陣列
元素	element	陣列裡的單一格子，如 `arr[2]`	程式語言一	陣列
初始化	initialization	宣告時給初值；`={0}` 整個補 0	程式語言一	陣列
越界	out of bounds	索引超出 `0 ~ size-1`，未定義行為（UB）	程式語言一	陣列
陣列退化	array decay	陣列傳給函式時變成指向首元素的位址	程式語言一	陣列
遍歷	traversal	用迴圈逐一走訪每個元素	程式語言一	陣列
線性搜尋	linear search	從頭找到尾，O(n)	程式語言一	陣列
二分搜尋	binary search	在**已排序**陣列對半找，O(log n)	程式語言一	陣列

§ 2　核心概念

核心概念

要存 100 個成績，宣告 100 個變數不可能。**陣列**把「一排同型態的資料」用一個名字管理：記憶體裡**連續排列**，用**索引**（從 0 起）存取。

```
int scores[100];      // 一行取代 100 個變數，用 scores[i] 存取
```

三個關鍵：① 索引**從 0 開始**，`arr[5]` 有 5 個元素、合法索引是 `0~4`；② 大小在**宣告時就固定**、必須是常數，不能中途改；③ 陣列和**迴圈**天生一對，`for (int i=0;i<n;i++)` 是遍歷的標準寫法。

§ 3 主要內容

3.1 宣告與初始化

```
int a[5];                          // 宣告 5 個 int（未初始化＝垃圾值）
int a[5] = {10, 20, 30, 40, 50}; // 宣告 + 初始化
int a[5] = {10, 20};               // 前兩個給值，其餘自動補 0
int a[5] = {0};                    // 全部初始化為 0
int a[]  = {10, 20, 30};           // 省略大小，編譯器算出＝3
```

3.2 陣列在記憶體：連續的格子

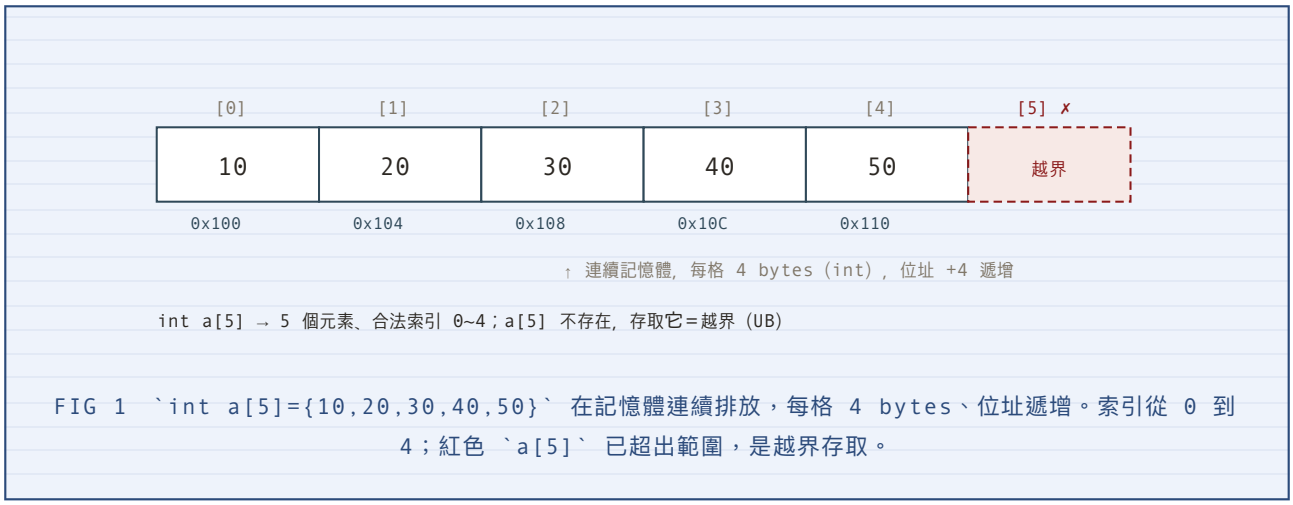

FIG 1 `int a[5]={10,20,30,40,50}` 在記憶體連續排放，每格 4 bytes、位址遞增。索引從 0 到 4；紅色 `a[5]` 已超出範圍，是越界存取。

3.3 存取、修改、遍歷

```
int a[5] = {10, 20, 30, 40, 50};
printf("%d\n", a[0]);              // 10（第一個）
printf("%d\n", a[4]);              // 50（最後一個）
a[2] = 99;                         // 修改第三個

int sum = 0;
for (int i = 0; i < 5; i++) sum += a[i];   // 遍歷求和 → 150
```

3.4 越界：C 不檢查邊界

```
int a[5] = {10,20,30,40,50};
a[5]  = 99;    // 越界！寫到不屬於你的記憶體
a[-1] = 88;    // 也是越界
```

C **不檢查陣列邊界**：越界不會編譯錯誤，也不一定當場 crash，可能改到別的變數、或「看起來正常」。這是最危險的 bug，緩衝區溢位攻擊正是利用它。本地用 `-fsanitize=address` 可在執行時抓到越界。

3.5 陣列傳給函式：傳的是位址

```c
void fill(int arr[], int n) {     // 不必寫大小；arr 退化成位址
    for (int i = 0; i < n; i++) arr[i] = i * i;
}
int main(void){
    int a[5];
    fill(a, 5);                   // 函式真的改到原陣列（傳址，非複製）
    return 0;
}
```

陣列名就是「指向第一個元素的位址」，傳給函式時**退化（decay）**成位址，所以函式內改動會**改到原陣列**（和變數的傳值不同，見 [[函式]]）。代價：函式內 `sizeof(arr)` 變成位址大小、**不是陣列大小**，所以要**另外傳 n**。本地用 `sizeof(a)/sizeof(a[0])` 只在原宣告處有效。

3.6 用值當索引：頻率統計

```c
int cnt[10] = {0};
int data[] = {3,1,4,1,5,9,2,6,5,3,5};
for (int i = 0; i < 11; i++) cnt[data[i]]++;   // 把「值」當成 cnt 的索引
```

「用值當索引」是陣列最強的技巧之一：O(n) 就統計完頻率，也可拿來做「出現過沒」的去重旗標。

3.7 搜尋與排序

```c
// 線性搜尋 O(n)：從頭找到尾
int search(int arr[], int n, int t){
    for (int i = 0; i < n; i++) if (arr[i] == t) return i;
    return -1;
}
// 氣泡排序 O(n²)：相鄰比較、把大的往後「冒泡」
for (int i = 0; i < n-1; i++)
    for (int j = 0; j < n-1-i; j++)
        if (a[j] > a[j+1]) { int t=a[j]; a[j]=a[j+1]; a[j+1]=t; }
```

二分搜尋（binary search）在**已排序**陣列裡每次砍一半，O(log n)，100 萬筆只要約 20 次比較；前提是先排序。

§ 4 語法與函式速查

```
int a[5];                          // 宣告（未初始化＝垃圾值）
int a[5] = {0};                    // 全 0
int a[]  = {1,2,3};                // 編譯器算長度＝3
a[i]                               // 存取第 i 個（i: 0 .. size-1）
for (int i=0;i<n;i++) ...          // 遍歷標準式
sizeof(a)/sizeof(a[0])             // 元素個數（傳函式後失效）
void f(int a[], int n)             // 傳函式：傳址 + 另傳 n
```

§ 5 常見錯誤

常見錯誤

- 越界：`arr[n]` 不存在（最後一個是 `arr[n-1]`）；`arr[-1]` 也越界。
- 未初始化就讀：`int a[100];` 內容是垃圾值，需要時用 `={0}`。
- 陣列開太小：題目說 N ≤ 10000 卻只開 `int a[100]`。
- 傳函式忘了另傳 n：函式內 `sizeof(arr)` 是位址大小、不是陣列長度。
- 二分搜尋用在**沒排序**的陣列（結果錯）。先排再搜。
- off-by-one：遍歷寫 `i <= n` 會多讀一格（越界）。

§ 6 練習題

練習 1（一般題）：陣列追蹤

下列印什麼？

```
int a[4] = {1, 2, 3, 4};
a[0] = a[3];
a[3] = a[0] + a[1];
printf("%d %d %d %d\n", a[0], a[1], a[2], a[3]);
```

> **引導步驟**
>
> 1. 先做 `a[0]=a[3]`，此時 a[0] 變成多少？
> 2. 再做 `a[3]=a[0]+a[1]`，用**更新後**的 a[0]。

解答

`4 2 3 6`。`a[0]=a[3]=4`；接著 `a[3]=a[0]+a[1]=4+2=6`（a[0] 已是 4）。a[1]、a[2] 沒動。

練習 2（一般題）：頻率統計

跑完後 `cnt[2]` 是多少？整個 `cnt` 長怎樣？

```
int cnt[5] = {0};
int a[] = {2, 0, 2, 3, 1, 2, 4, 0};
for (int i = 0; i < 8; i++) cnt[a[i]]++;
```

> **引導步驟**
>
> 1. `cnt[a[i]]++` 表示「a[i] 這個值又出現一次」。
> 2. 數每個值出現幾次。

解答

`cnt[2] = 3`（2 出現在 a[0]、a[2]、a[5]）。整個 `cnt = {2, 1, 3, 1, 1}`（0 出現 2 次、1 一次、2 三次、3 一次、4 一次）。

練習 3（重要題）：抓 bug

下列想加總 5 個元素，為何危險？怎麼修？

```
int a[5] = {1, 2, 3, 4, 5};
int sum = 0;
for (int i = 0; i <= 5; i++) sum += a[i];
```

引導步驟

1. 迴圈條件 `i <= 5` 會讓 i 跑到多少？
2. `a[5]` 存在嗎？

解答

`i <= 5` 讓 i 跑到 5，存取 `a[5]` 是**越界**（合法索引只到 4）。讀到的是隔壁記憶體的垃圾值，sum 結果不可靠，甚至可能 crash（UB）。修：把條件改成 `i < 5`（或 `i <= 4`）。

```
for (int i = 0; i < 5; i++) sum += a[i];
```

易錯

· 遍歷用 `<=` 造成 off-by-one 越界。陣列遍歷一律 `i < n`。

§ 7 自我檢核

- ☐ 會宣告/初始化陣列，知道大小固定、索引從 0 起。
- ☐ 能畫出陣列在記憶體的連續排列（每格 4 bytes、位址 +4）。
- ☐ 會用迴圈遍歷、求和、找最大最小。
- ☐ 知道越界是未定義行為、C 不檢查邊界（本地用 ASan 抓）。
- ☐ 懂陣列傳函式是傳址（改得到原陣列），且要另傳 n、sizeof 會失效。
- ☐ 會用「值當索引」做頻率統計與去重。
- ☐ 分得清線性 O(n) 與二分 O(log n)，知道二分要先排序。