



# 淺談。陣列

1111資訊社



# 如果想要紀錄1個人的成績？

可以宣告1個整數來記錄成績。

```
int score = 100;
```

# 如果想要紀錄5個人的成績？

可以宣告5個整數來記錄成績。

```
int score1 = 100, score2 = 89, score3 = 74,  
    score4 = 70, score5 = 40;
```

# 如果想要紀錄100個人的成績？

可以宣告100個整數來記錄成績。

```
int score1 = 1, score2 = 4, score3 = 37,  
    score4 = 73, score5 = 3, score6 = 30,  
    score7 = 32, score8 = 35, score9 = 52,  
    score10 = 57, score11 = 48, score12 = 72,  
    score13 = 50, score14 = 59, score15 = 100,
```

以下省略...

```
score19 = 89, score20 = 34, score21 = 89,
```

# 如果想要紀錄100個人的成績？

可以宣告100個整數來記錄成績。

```
int score1 = 1, score2 = 4, score3 = 37,  
    score4 = 73, score5 = 3, score6 = 30,  
    score7 = 10, score8 = 27, score9 = 52,  
    score10 = 57, score11 = 48, score12 = 72,  
    score13 = 50, score14 = 90, score15 = 20,  
    score16 = 22, score17 = 55, score18 = 84,  
    score19 = 89, score20 = 34, score21 = 89,
```

這樣效率也太低了吧！

有沒有更高效的方法呢？

陣列 Array

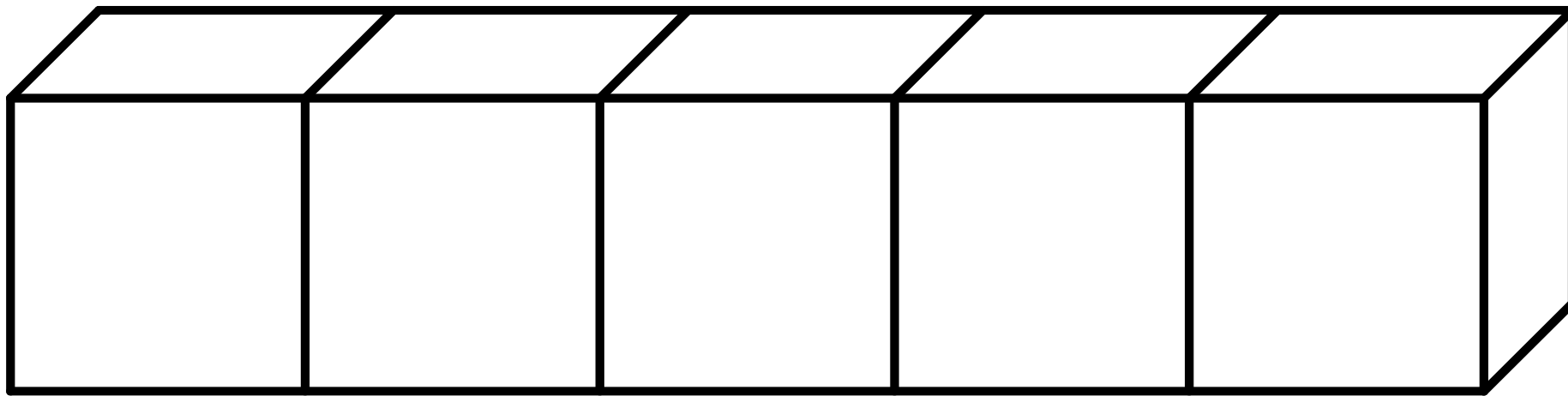
# 陣列 Array

在計算機科學中，**陣列資料結構**（英語：array data structure），簡稱**陣列**（英語：Array），是由相同類型的元素（element）的集合所組成的資料結構，分配一塊連續的記憶體來儲存。利用元素的索引（index）可以計算出該元素對應的儲存位址。最簡單的資料結構是線性陣列，也稱為一維陣列。

Source: Wikipedia Array\_(data\_structure)

# 陣列 Array

可以將陣列想像成連續的盒子（連續的記憶體空間）

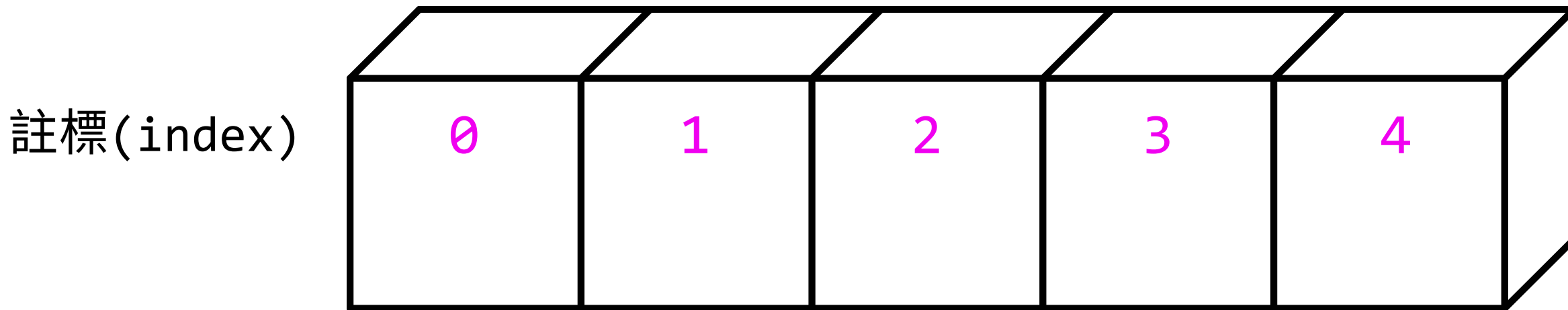


以長度為5的一維陣列為例



# 陣列 Array

可以將陣列想像成連續的盒子（連續的記憶體空間）  
盒外有編號，稱為註標(index) 從0開始算



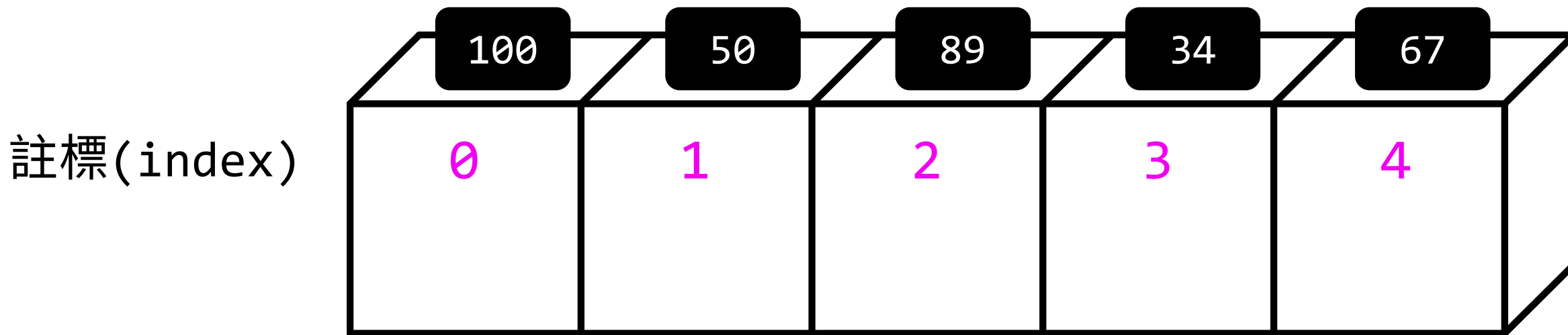
以長度為5的一維陣列為例

# 陣列 Array

可以將陣列想像成連續的盒子（連續的記憶體空間）

盒外有編號，稱為註標(index) 從0開始算

盒內有資料



以長度為5的一維陣列為例

# 陣列 Array

- 在C++中可以這樣宣告：

**類別名** 名稱[大小];

- 例如：

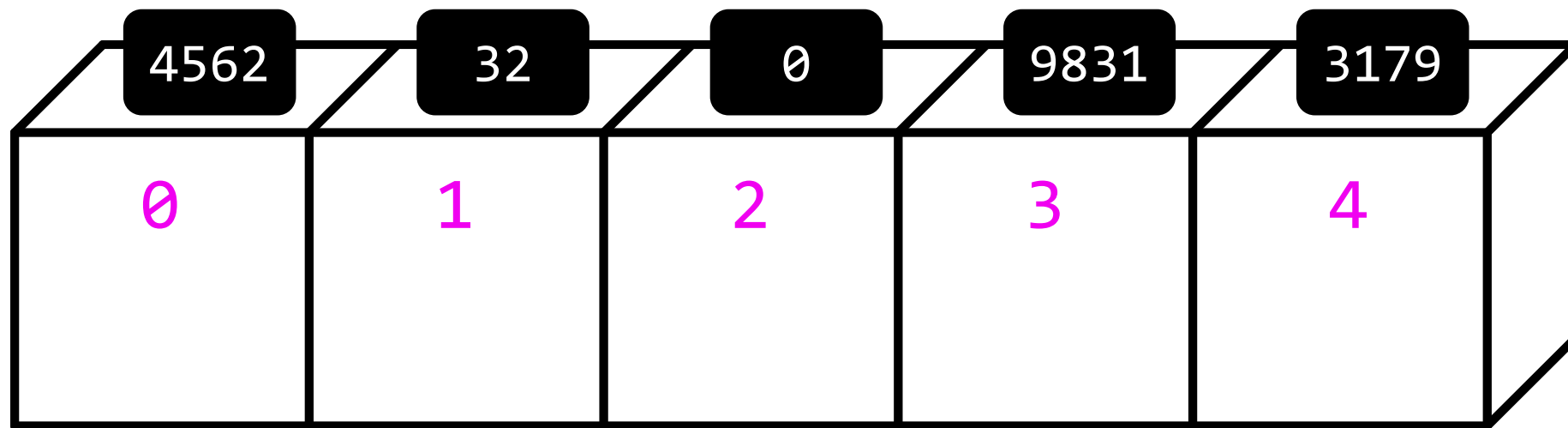
```
int score[5];  
float temperature[31];  
bool isPassed[10];  
string score[5];
```

# 陣列 Array

- 當陣列被宣告後，系統會在記憶體中找一塊連續的記憶體空間來存放陣列

宣告在區域變數的陣列，若沒有初始化，視編譯器而定，內部可能有「垃圾」（記憶體殘值）

註標(index)



以長度為5的一維陣列為例

# 陣列 Array

- 在C++中可以這樣初始化：

**類別名** 名稱[大小(可填可不填)] = {元素1, 元素2, ..., 元素n};

編譯器會自動推導

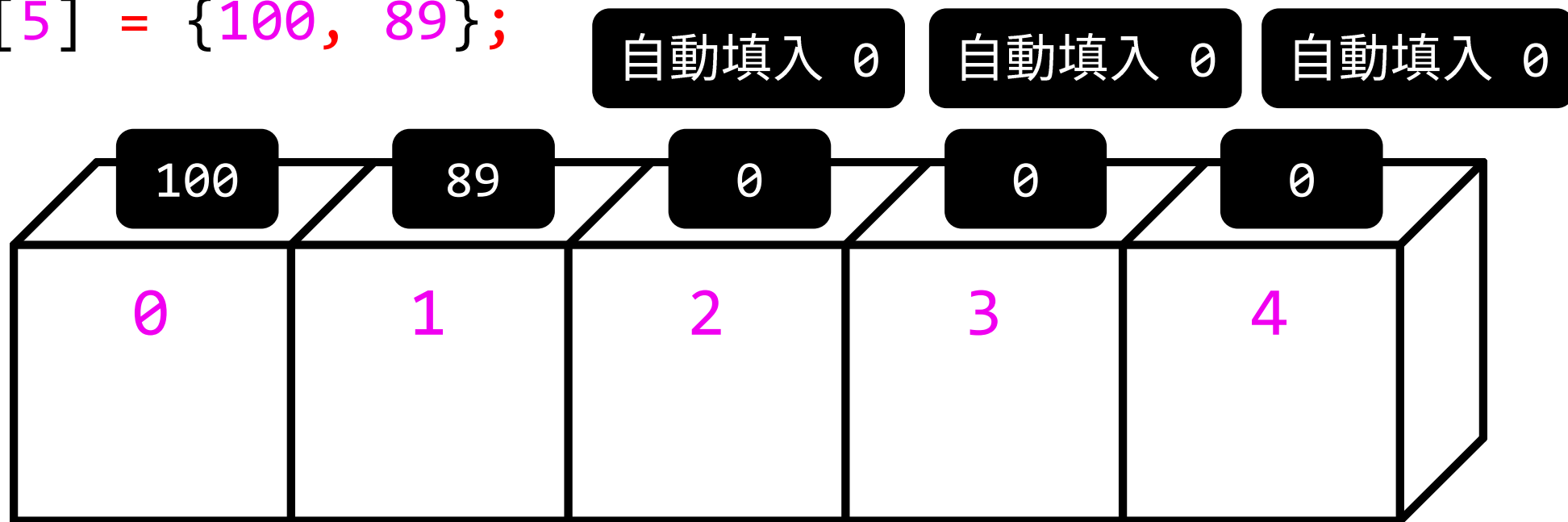
```
int score[] = {100, 89, 74, 70, 40};
```

# 陣列 Array

- 若 給定元素大小 < 陣列大小，則會自動填入該類別的預設值(0)

```
int score[5] = {100, 89};
```

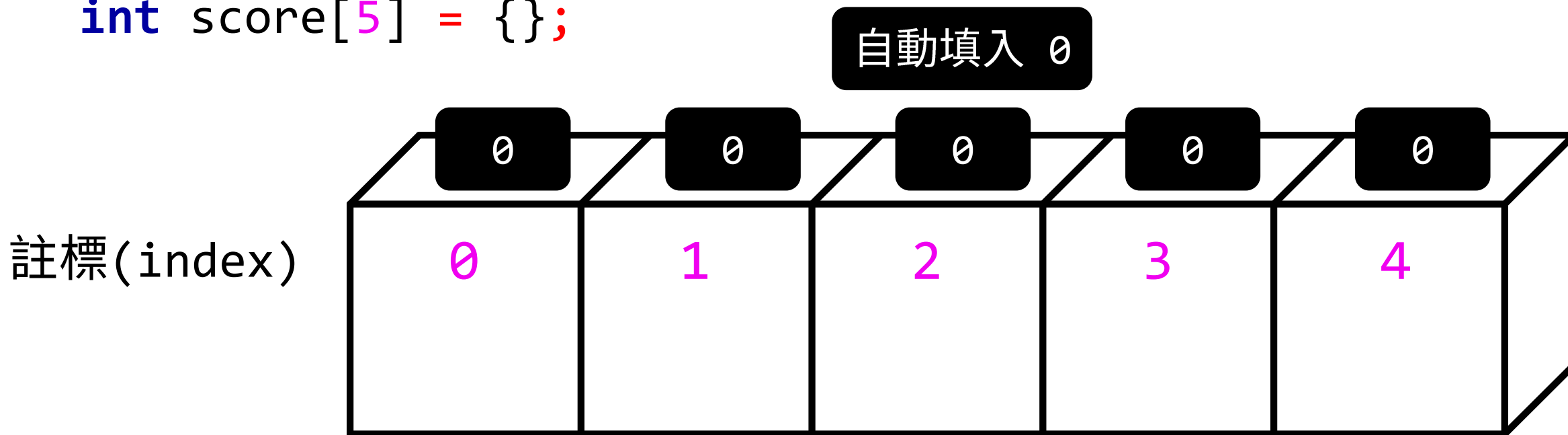
註標(index)



# 陣列 Array

- 若 給定元素大小 < 陣列大小，則會自動填入該類別的預設值(0)

```
int score[5] = {};
```



# 陣列 Array

- 存取陣列的元素：

名稱[註標(index)]

- 例如：

score[2]

temperature[15]

isPassed[0]



# 陣列 Array

- 搭配for迴圈，存取陣列裡的元素

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int score[] = {100, 89, 74, 70, 40};
7
8      for (int i = 0; i < 5; i++){
9          cout << score[i] << " ";
10     }
11
12     return 0;
13 }
```

宣告一陣列

讓i從0跑到4

輸出score中  
第i個元素

# 陣列 Array

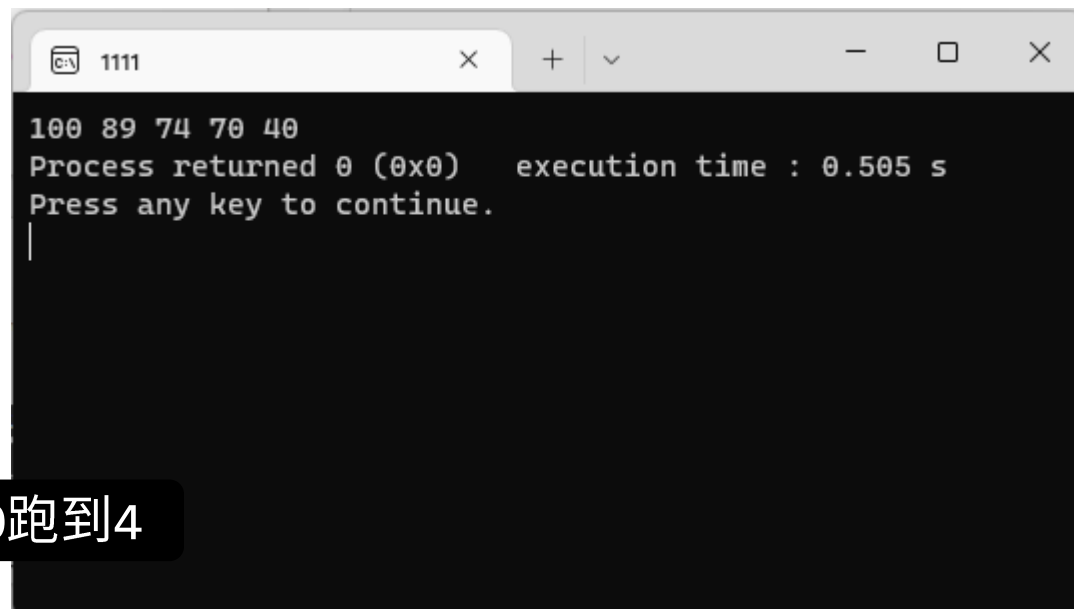
- 搭配for迴圈，存取陣列裡的元素

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int score[] = {100, 89, 74, 70, 40};
7
8      for (int i = 0; i < 5; i++){
9          cout << score[i] << " ";
10     }
11
12     return 0;
13 }
```

宣告一陣列

讓i從0跑到4

輸出score中  
第i個元素



```
1111
100 89 74 70 40
Process returned 0 (0x0)   execution time : 0.505 s
Press any key to continue.
```

# 如果要記錄5個人7個科目的成績？

可以宣告5個陣列來記錄7個科目每個人的成績。

```
int score1[5] = {100, 90, 94, 43, 67};  
int score2[5] = {45, 12, 0, 54, 67};  
int score3[5] = {67, 1, 43, 87, 91};  
int score4[5] = {50, 58, 32, 43, 0};  
int score5[5] = {99, 90, 100, 57, 67};  
int score6[5] = {98, 54, 18, 43, 67};  
int score7[5] = {30, 90, 94, 37, 67};
```

# 如果要記錄5個人100個科目的成績？

可以宣告5個陣列來記錄100個科目每個人的成績。

```
int score1[5] = {100, 90, 94, 43, 67};
```

```
int score2[5] = {45, 12, 0, 54, 67};
```

```
int score3[5] = {67, 1, 43, 87, 91};
```

```
int score4[5] = {50, 58, 32, 43, 0};
```

```
int score5[5] = {99, 90, 100, 57, 67};
```

```
int score6[5] = {98, 54, 18, 43, 67};
```

```
int score7[5] = {99, 90, 94, 57, 67};
```

```
int score8[5] = {50, 58, 32, 57, 67};
```

```
int score9[5] = {99, 90, 94, 57, 67};
```

以下省略...

# 如果要記錄5個人100個科目的成績？

可以宣告5個陣列來記錄100個科目每個人的成績。

```
int score1[5] = {100, 90, 94, 43, 67};
```

```
int score2[5] = {45, 12, 0, 54, 67};
```

```
int score3[5] = {67, 45, 12, 0, 91};
```

```
int score4[5] = {0, 67, 45, 12, 91};
```

```
int score5[5] = {99, 90, 100, 57, 67};
```

```
int score6[5] = {99, 90, 100, 57, 67};
```

```
int score7[5] = {30, 90, 94, 37, 67};
```

```
int score8[5] = {30, 90, 94, 37, 67};
```

```
int score9[5] = {30, 90, 94, 37, 67};
```

這樣效率也太低了吧！

有沒有更高效的方法呢？

多維陣列

# 多維陣列

只有1個註標的陣列為一維陣列，含有2個註標的陣列為二維陣列，含有3個註標的陣列為三維陣列，依此類推。

當註標多餘1時，我們稱該陣列為多維陣列。

# 如果要記錄5個人7個科目的成績？

可以宣告1個二維陣列來記錄7個科目每個人的成績。

```
int score[5][7] = {{100, 90, 94, 43, 67, 45, 12},  
                    {67, 1, 43, 87, 0, 54, 67},  
                    {67, 1, 43, 87, 91, 50, 58},  
                    {50, 58, 32, 43, 0, 90, 100},  
                    {99, 90, 100, 57, 67, 54, 18}};
```

5列7欄



# 如果要記錄5個人7個科目的成績？

```
int score[5][7] = {{100, 90, 94, 43, 67, 45, 12},  
                    {67, 1, 43, 87, 0, 54, 67},  
                    {67, 1, 43, 87, 91, 50, 58},  
                    {50, 58, 32, 43, 0, 90, 100},  
                    {99, 90, 100, 57, 67, 54, 18}};
```

5列7欄

score[3][4] <- 第3個人第4科目的成績

score[0][6] <- 第0個人第6科目的成績

score[1][0] <- 第1個人第0科目的成績

# 多維陣列

- 搭配for迴圈，存取陣列裡的元素

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int score[5][7] = {{100, 90, 94, 43, 67, 45, 12},
7                          {67, 1, 43, 87, 0, 54, 67},
8                          {67, 1, 43, 87, 91, 50, 58},
9                          {50, 58, 32, 43, 0, 90, 100},
10                         {99, 90, 100, 57, 67, 54, 18}};
11
```

5列7欄，  
第一格註標值代表人，  
第二格註標值代表科目

宣告一陣列

讓i從0跑到4

讓j從0跑到6

```
for (int i = 0; i < 5; i++){
    for (int j = 0; j < 7; j++){
        cout << score[i][j] << "\t";
    }
    cout << "\n";
}

return 0;
```

輸出score中第i個元素的第j個元素

# 多維陣列

- 搭配for迴圈，存取陣列裡的元素

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int score[5][7] = {{100, 90, 94, 43, 67, 45, 12},
7                          {67, 1, 43, 87, 0, 54, 67},
8                          {67, 1, 43, 87, 91, 50, 58},
9                          {50, 58, 32, 43, 0, 90, 100},
10                         {99, 90, 100, 57, 67, 54, 18}};
11
```

宣告一陣列

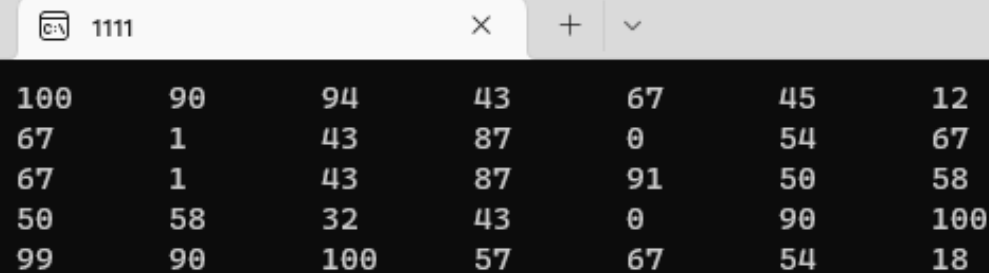
5列7欄，  
第一格註標值代表人，  
第二格註標值代表科目

讓i從0跑到4

讓j從0跑到6

```
12  for (int i = 0; i < 5; i++){
13      for (int j = 0; j < 7; j++){
14          cout << score[i][j] << "\t";
15      }
16      cout << "\n";
17  }
18
19  return 0;
20
```

輸出score中第i個元  
素的第j個元素



100	90	94	43	67	45	12
67	1	43	87	0	54	67
67	1	43	87	91	50	58
50	58	32	43	0	90	100
99	90	100	57	67	54	18

Process returned 0 (0x0) execution time : 0.024 s  
Press any key to continue.