

5.3 多維陣列

二維陣列

把索引值擴展為 2 維，我們就可以得到二維陣列。

一個大小為 $m \times n$ 的二維陣列，可以這樣宣告。

```
// 宣告一個 4 x 6 的 int 二維陣列
int A[4][6];
```

和一維陣列一樣，可以在宣告時給定初值。

```
int A[4][6] = {
    {1, 2, 3, 4, 5, 6},
    {5, 12, 7, 11, 9, 8},
    {10, 21, 13, 22, 23, 16},
    {4, 78, 13, 45, 51, 11}
};
```

陣列 A

	[0]	[1]	[2]	[3]	[4]	[5]	
[0]	1	2	3	4	5	6	
[1]	5	12	7	11	9	8	-----> A[1][3]
[2]	10	21	13	22	23	16	
[3]	4	78	13	45	51	11	-----> A[3][5]

搭配雙層迴圈遍歷其值

我們可以使用雙層迴圈，把前面那個二維陣列的值印出來。

```
for(int i=0; i<4; i++) {
    for(int j=0; j<6; j++) {
        cout << A[i][j] << " ";
    }
}
```

```
    cout << endl;
}
```

1 2 3 4 5 6 5 12 7 11 9 8 10 21 13 22 23 16 4 78 13 45 51 11

練習：2D 地圖

給定一張 $m \times n$ 大小的地圖，以及各地貌的代表數字，請輸出該地圖。

\$ 1 \le m, n \le 100 \$

輸入說明：

第一行是兩個正整數 m n ，表示陣列的 列數(row)、行數(column)。

接下來是共 m 列，每列有 n 個整數的地圖資訊，表示該位置的地貌代碼。

接著是一個整數 k ，表示有 k 種地貌。

最後是 k 列，每列為一個整數 i 與一個字元 c ，表示代碼 i 的地貌為 c 。

輸出說明：

輸出該地圖的地貌，如範例輸出。

範例輸入：

3 4 1 1 1 1 2 2 0 1 1 2 0 1 3 0 _ 1 # 2 *

範例輸出：

**_# #*_#

```
#include <iostream>

using namespace std;

int main()
{
    int m, n;
    cin >> m >> n;

    int M[100][100]; // 這樣比較安全，若 size 太大可考慮宣告在全域區
    // int M[m][n]; // C99 的 VLA 允許這麼宣告，但是若 size 太大會有問題

    for(int i=0; i<m; i++) {
        for(int j=0; j<n; j++) {
            cin >> M[i][j];
        }
    }

    int k;
```

```

cin >> k;
int N[k];    // 地貌代碼
char T[k];   // 地貌

for(int i=0; i<k; i++) {
    cin >> N[i] >> T[i];
}

for(int i=0; i<m; i++) {
    for(int j=0; j<n; j++) {
        for(int u=0; u<k; u++) {
            if(M[i][j]==N[u]) {    // 在 N 中找出代碼 M[i][j] 的位置 u
                cout << T[u];    // 輸出地貌 T[u]
                break;
            }
        }
    }
    cout << endl;
}

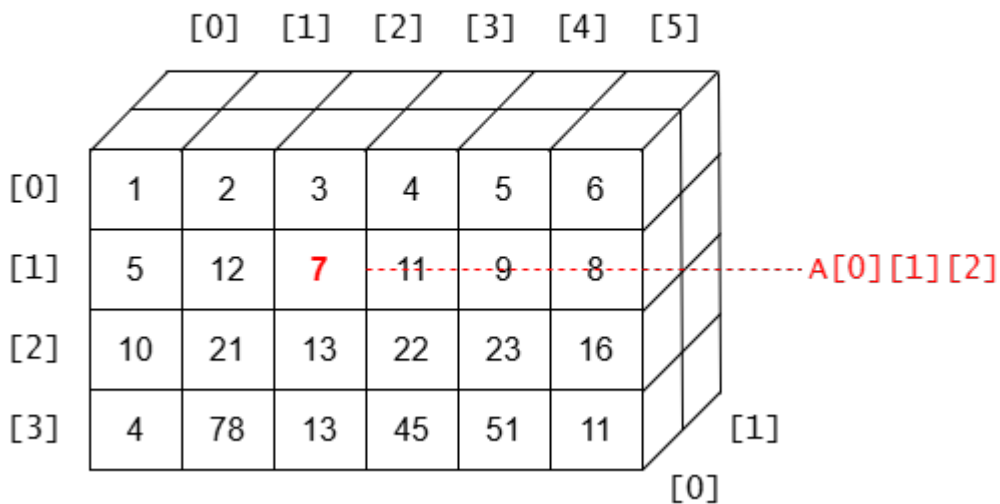
return 0;
}

```

多維陣列

如果把二維陣列想像成一個平面，那麼三維陣列就可以想像成一個長方體。

陣列 A



平常我們很少使用超過 3 維的陣列，除非你很確定自己需要，否則在你宣告一個大於 3 維的陣列之前，可以想一想，有沒有更好的方式可以不要用到這麼高維的陣列。

🔄Revision #13

★Created 2024-04-14 10:21:01 UTC by huihui

✎Updated 2025-12-22 07:52:12 UTC by huihui