1

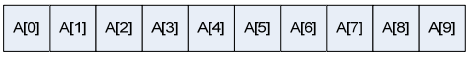
預告

- 期中考要來了! 期中考要來了 ...
  - 日期: 4/20 (Fri.) 上課時段
  - 地點: TR-413-2
  - 範圍: 考至一維陣列 (今日上課內容)
  - 題型: 選擇、填空、程式 (如果照慣例的話)

2

回顧

- 陣列
  - `int A[10];`



- 陣列初始化
  - `int A[] = {1,2,3,4,5};`
  - `int B[10] ={1,2,3,4,5};`
- C 字串 / 一維字元陣列

3

迴圈與陣列

- 請寫一程式，讓使用者輸入 10 個整數並儲存至陣列中，之後將此 10 個整數由小排至大後印出來。

```
請輸入第1個數字:1
請輸入第2個數字:10
請輸入第3個數字:2
請輸入第4個數字:9
請輸入第5個數字:3
請輸入第6個數字:8
請輸入第7個數字:4
請輸入第8個數字:6
請輸入第9個數字:5
請輸入第10個數字:7
1 2 3 4 5 6 7 8 9 10
```

提示：
- 需使用雙層迴圈
- 外層迴圈控制下頁中的紅三角形位置
- **紅三角形位置從 0 開始，至 9 為止**
- 內層迴圈控制下頁中藍三角形位置
- **藍三角形由紅三角形右邊開始至 9 為止**
- 若紅三角形位置上的值大於藍三角形位置上的值時，則將其內容交換！

4

```
#include <iostream>
using namespace std;
int main() {
    int a[10];

    for(int i=0;i<10;++i) {
        cout << "請輸入第" << i+1 << "個數字:";
        cin >> a[i];
    }
    for(int i=0;i<9;++i) {
        for(int j=i+1;j<10;++j) {
            if(a[i] < a[j]) continue;
            int tmp = a[i];
            a[i] = a[j];
            a[j] = tmp;
        }
    }
    for(int i=0;i<10;++i) {
        cout << a[i] << " ";
    }
    return 0;
}
```

5

Lecture 08

多維陣列

6

Outline

- 二維與多維陣列
- C-字串陣列 / 字元的二維陣列
- 向量與矩陣運算

7

二維陣列與多維陣列

8

一維陣列

- 陣列為多個**同一型態**變數之組合
- int a[10];
  - 可存放 10 個整數資料的陣列，可視為 10 個變數排成一個隊伍，並給每一個變數一個編號，自 0 號開始、至 9 號為止。
  - a[0], a[1], a[2], a[3], ..., a[9]

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

9

二維陣列的宣告

- 陣列為多個**同一型態**變數之組合
- int A**[3]**[10];
  - 可存放 **30** 個整數資料的二維陣列，可視為 **30** 個變數排成 **3** 個隊伍 (**3列**)，並給每一個變數兩個編號，一個編號指定在第幾個隊伍、另一個編號指定在隊伍內的順序。
  - A[0][0], A[0][1], A[0][2], ... A[0][9], A[1][0], A[1][1], ...

A[0][0]	A[0][1]	A[0][2]	A[0][3]	A[0][4]	A[0][5]	A[0][6]	A[0][7]	A[0][8]	A[0][9]
A[1][0]	A[1][1]	A[1][2]	A[1][3]	A[1][4]	A[1][5]	A[1][6]	A[1][7]	A[1][8]	A[1][9]
A[2][0]	A[2][1]	A[2][2]	A[2][3]	A[2][4]	A[2][5]	A[2][6]	A[2][7]	A[2][8]	A[2][9]

10

二維陣列 (Array) 的使用

- int cars[3][6];
  - 宣告一整數陣列, 名為 cars, 可存放3列6行共 18 個整數型別的資料
- 使用
  - cars[0][1] = 3;→ 將 3 存入 cars 陣列中的第 0 列第1行的元素
  - cars[2][0]++; → 將 cars 陣列中的第2列第0行的元素增加 1
  - cars[1][0]=cars[1][1]+a; → 將cars陣列中的第1列第1行的值取出，與變數 a 作加法運算後，將結果存入第1列第0行的元素。
- 為什麼有二維陣列或多維陣列？
  - 對某些運算來說很自然 (e.g. 行列式、矩陣) ...

11

8-1.cpp

```
int a[3][5];

for(int i=0;i<3;i++) {
    for(int j=0;j<5;j++) {
        a[i][j] = i*10 + j;
    }
}

for(int i=0;i<3;i++) {
    for(int j=0;j<5;j++) {
        cout << a[i][j] << " ";
    }
    cout << "\n";
}
```

```
0 1 2 3 4
10 11 12 13 14
20 21 22 23 24
```

12

二維陣列的初始化

- `int a[3][4] = {{1,2,3,4}, {5,6,7,8}, {9,10,11,12}};`

陣列元素排列方式

A[0][0]	A[0][1]	A[0][2]	A[0][3]
A[1][0]	A[1][1]	A[1][2]	A[1][3]
A[2][0]	A[2][1]	A[2][2]	A[2][3]

各陣列元素之初始值

1	2	3	4
5	6	7	8
9	10	11	12

13

請想想以下陣列初始化後之內容為?

- `int a[3][4] = {1,2,3,4};`
- `int a[3][4] = {{1,2,3,4}, {8, 1}, {4, 2}};`
- `int a[3][4] = {{0},{1,2,3,4},{5,1,2,3}};`
- `int a[3][4] = {1,2,3,4,5,6,7,8};`
- `int a[][4] = {{1, 2}, {3, 4, 5}};`

1 2 3 4 0 0 0 0 0 0 0 0	1 2 3 4 8 1 0 0 4 2 0 0	0 0 0 0 1 2 3 4 5 1 2 3	1 2 3 4 5 6 7 8 0 0 0 0	1 2 0 0 3 4 5 0

在陣列有給初始值時，且陣列大小被省略時，編譯器會根據初始值的個數來決定陣列大小。但是只有第一個維度可以省略！ `double A[][4], b[], c[][10][40];` 而必需有給初始值時才能省略第一個維度大小。

14

8-2.cpp

```
#include <iostream>
using namespace std;
int main() {
    const int n = 3;
    double a[n][n] = {
        {1,2,3},
        {4,5,6},
        {7,8,9}
    };
    for(int col=0;col<n;++col) {
        cout << "第 " << col+1 << " 行 (column) 向量為:";
        for(int row=0;row<n;++row) {
            cout << a[row][col] << " ";
        }
        cout << "\n";
    }
```

```
第 1 行 (column) 向量為:1 4 7
第 2 行 (column) 向量為:2 5 8
第 3 行 (column) 向量為:3 6 9
第 1 列 (row) 向量為:1 2 3
第 2 列 (row) 向量為:4 5 6
第 3 列 (row) 向量為:7 8 9
```

15

8-2.cpp

```
    for(int row=0;row<n;++row) {
        cout << "第 " << row+1 << " 列 (row) 向量為:";
        for(int col=0;col<n;++col) {
            cout << a[row][col] << " ";
        }
        cout << "\n";
    }

    return 0;
}
```

```
第 1 行 (column) 向量為:1 4 7
第 2 行 (column) 向量為:2 5 8
第 3 行 (column) 向量為:3 6 9
第 1 列 (row) 向量為:1 2 3
第 2 列 (row) 向量為:4 5 6
第 3 列 (row) 向量為:7 8 9
```

16

多維陣列

- 二維以上陣列 (稱為多維陣列) 在實務上應用比較少。
- 宣告
  - int a[10][5][3];
    - 宣告一三維陣列，第一維度有10個、第二維度5個、第三維度3個。 → 共可儲存150個不同的整數值
    - 可想像成我們宣告了 10 個不同的 5 列 3 行之二維陣列
    - 亦可想像成一 10 層樓的房子、每層樓裡有 5 列 3 行個房間。
  - float q[100][200][10][5];
    - 宣告一個四維陣列，共可儲存 100 x 200 x 10 x 5 個不同的浮點數。
    - 可想像為我們有 100 棟房子，每一棟樓高為200層，每一層裡有10列5行個房間

17

C-字串陣列 / 字元的二維陣列

18

回顧

- 字串 = 字元的一維陣列
  - `char a[10] = "ABCDEFGHI";`
- `cin` 可以從鍵盤讀取字串
  - `char a[10];`
  - `cin >> a;`
- `cout` 可以將 C字串輸出
  - `char a[]="Hello";`
  - `cout << a;`

19

8-3.cpp

```
This is a test.
請輸入一串字: Hello Kitty
Hello
```

cin 利用空白或是換行符號來分隔不同資料，所以只能藉此得到不含空白的字串。

```
#include <iostream>
using namespace std;

int main() {
  char msg[200] = "This is a test.";
  cout << msg;

  cout << "\n請輸入一串字: ";
  cin >> msg;
  cout << msg;

  return 0;
}
```

20

二維字元陣列 / 字串陣列

- char a[][6] = {{"Pig"}, {"Tiger"}, {"Dog"}};
- char a[3][6] = {{"Pig"}, {"Tiger"}, {"Dog"}};
- char [3][6] =
  {{'P', 'i', 'g', 0}, {'T', 'i', 'g', 'e', 'r', 0},
  {'D', 'o', 'g', 0}};

'P'	'i'	'g'	0		
'T'	'i'	'g'	'e'	'r'	0
'D'	'o'	'g'	0		

21

8-4.cpp

```
#include <iostream>
using namespace std;

int main() {
  char a[][6] = {{"Pig"}, {"Tiger"}, {"Dog"}};
  int i;

  for(i=0;i<3;i++) {
     cout << a[i] << endl;
  }

  return 0;
}
```

```
Pig
Tiger
Dog
```

22

8-5.cpp

```
#include <iostream>
using namespace std;

int main() {
    const int n = 30;
    char canvas[n][n];
    for(int i=0;i<n;++i) {
        for(int j=0;j<n;++j) {
            canvas[i][j] = ' ';
        }
    }
```

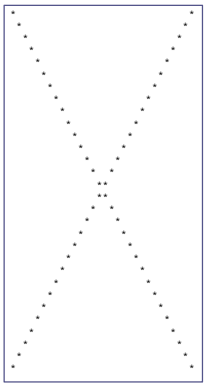
23

8-5.cpp

```
    for(int i=0;i<n;++i) {
        canvas[i][i] = '*';
        canvas[n-i-1][i] = '*';
    }

    for(int i=0;i<n;++i) {
        for(int j=0;j<n;++j) {
            cout << canvas[i][j];
        }
        cout << "\n";
    }

    return 0;
}
```



24

向量與矩陣運算

25

向量 (vectors)

- 在電腦裡，向量可以以一維陣列儲存與計算
- 以下範例計算 a 向量 (1,2,3) 與 b 向量 (4,5,6) 的外積

$$\vec{a}\times\vec{b}=\begin{bmatrix}\vec{i} & \vec{j} & \vec{k}\\ a_1 & a_2 & a_3\\ b_1 & b_2 & b_3\end{bmatrix}=(a_2b_3-a_3b_2)\vec{i}+(a_3b_1-a_1b_3)\vec{j}+(a_1b_2-a_2b_1)\vec{k}$$

- 那如果向量要算內積呢 (inner-dot/dot product)？

26

8-6.cpp

-3 6 -3

```
#include <iostream>
using namespace std;

int main() {
      double a[] = {1,2,3};
      double b[] = {4,5,6};
      double ans[3];

      ans[0] = a[1]*b[2] - a[2]*b[1];
      ans[1] = a[2]*b[0] - a[0]*b[2];
      ans[2] = a[0]*b[1] - a[1]*b[0];

      for(int i=0;i<3;++i) cout << ans[i] << " " ;

      return 0;
}
```

27

矩陣 (matrix)

- 矩陣在電腦裡，則很自然地以二維陣列儲存與運算

$$\begin{bmatrix}a_{11} & a_{12} & a_{13} & a_{14}\\ a_{21} & a_{22} & a_{23} & a_{24}\\ a_{31} & a_{31} & a_{33} & a_{34}\end{bmatrix}$$

- 矩陣與向量相乘

$$\begin{bmatrix}a_{11} & a_{12} & a_{13} & a_{14}\\ a_{21} & a_{22} & a_{23} & a_{24}\\ a_{31} & a_{31} & a_{33} & a_{34}\end{bmatrix}\begin{bmatrix}b_1\\ b_2\\ b_3\\ b_4\end{bmatrix}=\begin{bmatrix}a_{11}b_1+a_{12}b_2+a_{13}b_3+a_{14}b_4\\ a_{21}b_1+a_{22}b_2+a_{23}b_3+a_{24}b_4\\ a_{31}b_1+a_{32}b_2+a_{33}b_3+a_{34}b_4\\ a_{41}b_1+a_{42}b_2+a_{43}b_3+a_{44}b_4\end{bmatrix}=\begin{bmatrix}c_1\\ c_2\\ c_3\\ c_4\end{bmatrix}$$

28

矩陣向量相乘

$$\begin{bmatrix}1 & 2 & 3 & 4\\ 1 & 0 & 1 & 0\\ 2 & 1 & 0 & 1\\ 0 & 1 & 0 & 1\end{bmatrix}\begin{bmatrix}4\\ 3\\ 2\\ 1\end{bmatrix}=\begin{bmatrix}1\times4+2\times3+3\times2+4\times1\\ 1\times4+0\times3+1\times2+0\times1\\ 2\times4+1\times3+0\times2+1\times1\\ 0\times4+1\times3+0\times2+1\times1\end{bmatrix}$$

$$c_i=\sum_{j=1}^{4}a_{ij}b_j \qquad c_1=a_{i1}b_1+a_{i2}b_2+a_{i3}b_3+a_{i4}b_4$$

29

8-7.cpp

```
int a[4][4] = {
    {1,2,3,4},
    {1,0,1,0},
    {2,1,0,1},
    {0,1,0,1}
};
int b[] = {4,3,2,1}, c[4]={0};
```

$$\begin{bmatrix}1 & 2 & 3 & 4\\ 1 & 0 & 1 & 0\\ 2 & 1 & 0 & 1\\ 0 & 1 & 0 & 1\end{bmatrix}\begin{bmatrix}4\\ 3\\ 2\\ 1\end{bmatrix}$$

30

8-7.cpp

```
for(int i=0;i<4;i++) {
    for(int j=0;j<4;j++) {
      c[i] += a[i][j] * b[j];
    }
}

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

$$c_i=\sum_{j=1}^{4}a_{ij}b_j \qquad c_1=a_{i1}b_1+a_{i2}b_2+a_{i3}b_3+a_{i4}b_4$$

31

隨堂練習

32

請修改8-5.cpp，以產生如右的圖形。