1

Lecture 8

C 字串 / 一維字元陣列
二維與多維陣列

2

一維字元陣列 / C -字串

- 字元陣列就是一種陣列，只是它的變數型態是字元 (char)
- char a[100];
  - 可用來存放 100 個字元的一維字元陣列。
  - `a[0] = 'a';` ← → `a[0] = 97;`
  - `a[2] = 65;` ← → `a[2] = 'A';`
- 何謂字元 (character)
  - 它其實是一種整數，但它的意義是給人看的「符號」，所以它的輸入輸出有著特別的處理機制。
  - 每個電腦上看到的「符號」背後都一編碼標準
    - 英文字母: ASCII (American Standard for Information Interchange)
    - 正體中文：Big-5、UTF-8 (萬國碼)
    - 簡體中文 ：GB-2312

3

8-1.cpp

```
#include <iostream>
using namespace std;
int main() {
  unsigned char i;
  int j;

  for(i=32;i<128;i++) {
      j=i;
      cout << j << ": " << i << endl;
  }
  return 0;
}
```

```
32:          48: 0
33: !        49: 1
34: "        50: 2
35: #        51: 3
36: $        52: 4
37: %        53: 5
38: &        54: 6
39: '        55: 7
40: (        56: 8
41: )        57: 9
42: *        58: :
43: +        59: ;
44: ,        60: <
45: -        61: =
46: .        62: >
47: /        63: ?
             .
             .
             .
```

4

其中有用的幾個控制碼，在 C/C++ 中以跳脫字元 (Escape sequence) 方式輸入：

\n: new line (換行)
\b: back space (擦去前一個字)
\r: carriage return (回到列開頭)
\a: alert (警告音)
\t: horizontal tab (水平定位)

\": 雙引號
\': 單引號
\\: backslash (反斜線)
\0: null character
\xhh: 以 16 進位方式指定字碼
(e.g. \x41 → 'A')

\v: vertical tab (垂直定位)
\f: form feed (換頁)

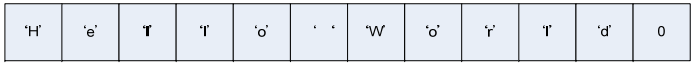
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	⌂

http://www.sciencelobby.com/ascii-table/ascii-table.html

5

一維字元陣列 / C -字串

- C- 字串 (C-string)
  - 在 C 語言中的 C 字串即為一連串的字元並以值 0 作為結尾的標記 (mark)，或稱為 null-terminated string.
  - 程式裡字串以雙引號包起來。
  - cout << "Hello World" << endl;
  - 以上之 Hello World 即為一字串。



6

8-2.cpp - C/C++ 對字串輸出的特別處理

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

int main() {
  char a[]= "Hello World";

  cout << a << endl;

  return 0;
}
```

```
Hello World
```

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

- 在此以 C 字串的方式對 a 這個字元陣列進行初始化的動作。也只有字元陣列與 string 型別的變數可以這樣作初始化。
- int a[] = "Hello World!"; ← 錯誤
- 因為 a 這個陣列是字元型別陣列，所以可以這樣把整個陣列內容印出來。

7

8-3.cpp

```
#include <iostream>
using namespace std;
int main() {
  char a[]= "Hello World";
  int i=0;
  while(a[i]!=0) {
      cout << i << ":" << a[i] << endl;
      i++;
  }
  a[4]=0; // a[4]='\0';
  cout << a << endl;
  return 0;
}
```

```
0:H
1:e
2:l
3:l
4:o
5:
6:W
7:o
8:r
9:l
10:d
Hell
```

8

8-4.cpp

```
char a[100], tmp;
int pos=0;

cout << "請輸入一字串: ";
cin >> a;

do {
    pos++;
    tmp = a[pos];
    a[pos] = 0;
    cout << a << endl;
    a[pos] = tmp;
} while(a[pos] != 0);
```

C/C++ 對字串輸入的特別處理要注意陣列需有足夠的空間！

```
請輸入一字串: 1234567890
1
12
123
1234
12345
123456
1234567
12345678
123456789
1234567890
```

9

小結

- C 字串 = 字元的一維陣列 != C++ 的字串

```
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;
```

10

C 字串的注意事項

- 注意 C 字串不能用指定運算子 (c = “Hello World”) 設定字串內容、也不能用來作比較判斷 (==) 來判斷兩字串內容是否相等。
  - 但之前介紹的 string 型別 (C++ 字串) 可以!
- 這些動作待我們未來介紹函式時才有辦法對 C 字串作這些操作。

```
char a[200];
a = "Hello Kitty";
```

```
char a[200];
if(a == "Hello Kitty") {
  ...
}
```

11

二維陣列與多維陣列

12

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 (索引)，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]

13

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) ...

14

8-5.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
```

15

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

16

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。

17

8-6.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";
    }
    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
```

18

8-7.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 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 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
```

19

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

20

C-字串陣列 / 字元的二維陣列

21

二維字元陣列 / C - 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		

22

8-8.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
```

23

8-9.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] = ' ';
        }
    }
```

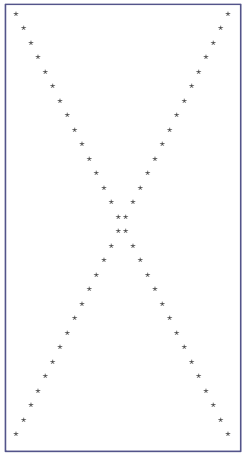
24

8-9.cpp (cont'd)

```
    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;
}
```



25

向量與矩陣運算

26

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)？

27

8-10.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;
}
```

28

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

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

29

矩陣向量相乘

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

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

30

8-11.cpp

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

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

31

8-11.cpp (cont'd)

```
for(int i=0;i<3;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] << " ";
}
```

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

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

32

那，矩陣相乘呢？

$$\begin{bmatrix}a_{11}&a_{12}&a_{13}&a_{14}\\a_{21}&a_{22}&a_{23}&a_{24}\\a_{31}&a_{32}&a_{33}&a_{34}\\a_{41}&a_{42}&a_{43}&a_{44}\end{bmatrix}\begin{bmatrix}b_{11}&b_{12}&b_{13}&b_{14}\\b_{21}&b_{22}&b_{23}&b_{24}\\b_{31}&b_{32}&b_{33}&b_{34}\\b_{41}&b_{42}&b_{43}&b_{44}\end{bmatrix}=\begin{bmatrix}c_{11}&c_{12}&c_{13}&c_{14}\\c_{21}&c_{22}&c_{23}&c_{24}\\c_{31}&c_{32}&c_{33}&c_{34}\\c_{41}&c_{42}&c_{43}&c_{44}\end{bmatrix}$$

$$c_{ij}=\sum_{k=1}^{4}a_{ik}b_{kj}$$

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

33

隨堂練習

34

字串的一維陣列 (字元的二維陣列)

- 在程式一開始宣告兩字串陣列 noun, verb 並初始化如下：
  - char noun[][4]={"狗", "豬", "雞", "羊"}; // 可以自己擴充
  - char verb[][4]={“咬”, “打”, “追”, “吃”}; // 可以自己擴充
- 其中 noun 為名詞，verb 為及物動詞。請利用程式列印出所有可能的名詞 + 及物動詞 + 名詞的組合。(利用三層巢狀迴圈)

狗咬狗	狗追狗
狗咬豬	狗追豬
狗咬雞	狗追雞
狗咬羊	狗追羊
狗打狗	.
狗打豬	.
狗打雞	.
狗打羊	

作業七

Due: 4/28/2017

36

請修改8-9.cpp，讓使用者選擇1-3以分別產生如下三種圖形 (都必需以迴圈完成)。

(1) (2) (3)