

預告

- 期中考要來了! 期中考要來了 ...
 - 日期: 4/13 or 4/20 (Fri.) 上課時段 (等下表決)
 - 地點: 另行公告
 - 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)

回顧

- 迴圈
 - `for(____; ____; ____){ ... }`
 - `while(____){ ... }`
 - `do { ... } while(____);`
 - `break; continue;`
- 無窮迴圈
 - `for(;;){ ... }, while(1){...}, do { ... } while(1);`

回顧

- 巢狀迴圈 – 迴圈裡還有迴圈 ...
 - e.g. 我們想要印出 7 列的米字號，每一列有 3 個米字號 (共有幾個米字號?)

```
for(int i=0;i<7;i++) {
    for(int j=0;j<3;j++) {
        cout << "***";
    }
    cout << endl;
}
```

Lecture 7

陣列

今日內容

- 一維陣列
- 陣列初始化
- C 字串 / 一維字元陣列
- 二維陣列與多維陣列

一維陣列

One-dimensional Array

一維陣列 (One-dimensional Array)

- 變數 (variable)
 - 名稱 – 用來區別不同的資料
 - 當我們需要儲存 20 個資料時需要 20 不同的變數名稱
 - 如果我們需要儲存 200 個資料的時候呢? 20000 個呢?
 - 型態 – int, double, short, long, float, ...
- 當一個程式被設計需要處理不同資料量時，通常無法只透過變數作運算，因為需要儲存的資料數量不同 → 這時可使用陣列來協助。
 - e.g. 成績計算程式，需要考慮班級大小

陣列 (Array) 的宣告

- 陣列為多個**同一型態**變數之組合，且此組合佔據連續的記憶體區塊
- 陣列的宣告
 - 變數型態 陣列名稱 [陣列大小];
 - char name [200];**
 - 宣告一陣列，可存放 200 個字元型別的資料
 - name[0], name[1], name[2], name[3], ..., name[199]
 - 此陣列具有 200 個元素 (elements)，每一個元素即可視為一個獨立的變數。
 - int cars [100];**
 - 宣告一整數陣列，可存放 100 個整數型別的資料
 - cars[0], cars[1], cars[2], ..., cars[99]
- 透過陣列宣告，我們可以快速得到大量的變數空間供程式使用

陣列 (Array) 的使用

- int A[10];**
 - 宣告一整數陣列，名稱為 A，可存放 10 個整數型別的資料
- | | | | | | | | | | |
|------|------|------|------|------|------|------|------|------|------|
| A[0] | A[1] | A[2] | A[3] | A[4] | A[5] | A[6] | A[7] | A[8] | A[9] |
|------|------|------|------|------|------|------|------|------|------|
- 使用
 - A[0] = 3; → 將 3 存入 A 陣列中的第 0 號元素
 - A[3]++; → 將 A 陣列中的 3 號元素增加 1
 - A[2]=A[4]+a;
 - 中括弧 [] 中的數字 (0, 3, 2, 4, ...) 用來選擇使用陣列中的那一個元素，被稱為索引 (index) 或下標 (subscript)
 - 常和 **for** 迴圈配合
 - 假設一陣列有 n 個元素，則可以**正確使用**的索引值為 0, 1, 2, ..., n-1

7-1.cpp

```

int main() {
    int data[15];

    for(int i=0;i<15;i++) {
        data[i] = 30-i*2;
    }
    for(int j=0;j<15;j++) {
        cout << j << ": " << data[j] << "\n";
    }
    for(int j=14;j>=0;j--) {
        cout << data[j] << " ";
    }

    return 0;
}

```

0:30
1:28
2:26
3:24
4:22
5:20
6:18
7:16
8:14
9:12
10:10
11:8
12:6
13:4
14:2
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30

7-2.cpp

```

int a[10];

for(int i=0;i<10;i++) {
    cout << "請輸入 " << i << " 號元素: ";
    cin >> a[i];
}

for(int i=9;i>=0;i--) {
    cout << a[i] << " ";
}

```

請輸入 0 號元素: 1
請輸入 1 號元素: 2
請輸入 2 號元素: 3
請輸入 3 號元素: 4
請輸入 4 號元素: 5
請輸入 5 號元素: 6
請輸入 6 號元素: 7
請輸入 7 號元素: 8
請輸入 8 號元素: 9
請輸入 9 號元素: 10
10, 9, 8, 7, 6, 5, 4, 3, 2, 1,

7-3.cpp

```

double a[10];

for(int i=0;i<10;i++) {
    cout << "請輸入 " << i << " 號元素: ";
    cin >> a[i];
}

for(int i=0;i<10;i++)
    cout << a[i] << " ";
cout << endl;

cout << a[0] << " ";
for(int i=1;i<10;i++) {
    cout << (a[i]+a[i-1]) * 0.5 << " ";
}

```

請輸入 0 號元素: 2
請輸入 1 號元素: 4
請輸入 2 號元素: 6
請輸入 3 號元素: 8
請輸入 4 號元素: 10
請輸入 5 號元素: 12
請輸入 6 號元素: 14
請輸入 7 號元素: 16
請輸入 8 號元素: 18
請輸入 9 號元素: 20
2 4 6 8 10 12 14 16 18 20
2 3 5 7 9 11 13 15 17 19

陣列初始化

陣列的初始化 (initialization)

- 變數在宣告/定義時，可給初始值
 - `int a=5, b=3;`
 - `double c=20;`
- 陣列亦可在宣告時，給定陣列中元素的**初始值**。

```

◦ int a[3] = {1, 2, 3};
◦ double b[5] = {1, 2, 3};
◦ float c[] = {1, 2, 3, 4};
◦ double d[10] = {0};
◦ char f[5] = {'H', 'E', 'L', 'L', 'O'}

```

1	2	3		
1.0	2.0	3.0	0.0	0.0
1.0	2.0	3.0	4.0	

7-4.cpp

```

#include <iostream>
using namespace std;
int main() {
    double a[20] = {1, 1};

    for(int i=2;i<20;i++) {
        a[i] = a[i-1] + a[i-2];
    }

    for(int i=0;i<20;i++) {
        cout << a[i] << " ";
    }
    return 0;
}
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765

```

7-5.cpp

```

#include <iostream>
using namespace std;
int main() {
    float a[10]={1.1f, 2.2f, 3.3f, 4.4f};

    for(int i=0;i<10;i++) {
        cout << a[i] << " ";
    }

    return 0;
}

```

1.1, 2.2, 3.3, 4.4, 0, 0, 0, 0, 0, 0,

只要陣列有進行初始化，則該陣列中未賦予初始值的元素會被給予 0。一般而言，陣列若沒有初始值，其內的數值未知。

小結

- 陣列讓我們的程式有儲存/處理大量資料的能力!

字元陣列 / C 字串

一維字元陣列 / C 字串

- 字元陣列即是陣列，只是它的變數型態是字元 (char)
- char a[100];
 - 可用來存放100個不同的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」。
 - 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)
 - 中文: Big-5

7-6.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: 0      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: ?
```

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

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

一維字元陣列 / C 字串

- 字串 (C-string)
 - 在 C 語言中的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.
 - 在程式裡寫的字串以雙引號包起來。
 - cout << "Hello World" << endl;
 - 以上之 Hello World 即為一字串。

'H'	'e'	'l'	'l'	'o'	' '	'W'	'o'	'r'	'l'	'd'	'0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

7-7.cpp

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

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

    cout << a << endl;

    return 0;
}
```

7-8.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
```

7-9.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);
```

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

C 字串的注意事項

- 注意 C 字串不能用在指定運算子 (=)、也不能用來作比較判斷 (==) 來判斷兩字串是否相等。
- 但之前介紹的 `string` 型別可以!
- 這些動作待我們未來介紹函式時才有辦法。

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

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

隨堂練習

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
• 內層迴圈控制下頁中藍三角形位置
• 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！
```

