

# 5. 陣列 & 字串

(使用C)

作者：劉宸均

為什麼需要陣列？

如果要記錄5個整數：

如果要記錄5個整數：

```
int a1;
```

```
int a2;
```

```
int a3;
```

```
int a4;
```

```
int a5;
```

10個呢？

# 10個呢？

int a1;      int a6;

int a2;      int a7;

int a3;      int a8;

int a4;      int a9;

int a5;      int a10;

200個呢？

# 200個呢？

int a1;	int a21;	int a41;	int a61;	int a81;	int a101;	int a121;	int a141;	int a161;	int a181;
int a2;	int a22;	int a42;	int a62;	int a82;	int a102;	int a122;	int a142;	int a162;	int a182;
int a3;	int a23;	int a43;	int a63;	int a83;	int a103;	int a123;	int a143;	int a163;	int a183;
int a4;	int a24;	int a44;	int a64;	int a84;	int a104;	int a124;	int a144;	int a164;	int a184;
int a5;	int a25;	int a45;	int a65;	int a85;	int a105;	int a125;	int a145;	int a165;	int a185;
int a6;	int a26;	int a46;	int a66;	int a86;	int a106;	int a126;	int a146;	int a166;	int a186;
int a7;	int a27;	int a47;	int a67;	int a87;	int a107;	int a127;	int a147;	int a167;	int a187;
int a8;	int a28;	int a48;	int a68;	int a88;	int a108;	int a128;	int a148;	int a168;	int a188;
int a9;	int a29;	int a49;	int a69;	int a89;	int a109;	int a129;	int a149;	int a169;	int a189;
int a10;	int a30;	int a50;	int a70;	int a90;	int a110;	int a130;	int a150;	int a170;	int a190;
int a11;	int a31;	int a51;	int a71;	int a91;	int a111;	int a131;	int a151;	int a171;	int a191;
int a12;	int a32;	int a52;	int a72;	int a92;	int a112;	int a132;	int a152;	int a172;	int a192;
int a13;	int a33;	int a53;	int a73;	int a93;	int a113;	int a133;	int a153;	int a173;	int a193;
int a14;	int a34;	int a54;	int a74;	int a94;	int a114;	int a134;	int a154;	int a174;	int a194;
int a15;	int a35;	int a55;	int a75;	int a95;	int a115;	int a135;	int a155;	int a175;	int a195;
int a16;	int a36;	int a56;	int a76;	int a96;	int a116;	int a136;	int a156;	int a176;	int a196;
int a17;	int a37;	int a57;	int a77;	int a97;	int a117;	int a137;	int a157;	int a177;	int a197;
int a18;	int a38;	int a58;	int a78;	int a98;	int a118;	int a138;	int a158;	int a178;	int a198;
int a19;	int a39;	int a59;	int a79;	int a99;	int a119;	int a139;	int a159;	int a179;	int a199;
int a20;	int a40;	int a60;	int a80;	int a100;	int a120;	int a140;	int a160;	int a180;	int a200;



# 200個呢？

int a1;	int a21;	int a41;	int a61;	int a81;	int a101;	int a121;	int a141;	int a161;	int a181;
int a2;	int a22;	int a42;	int a62;	int a82;	int a102;	int a122;	int a142;	int a162;	int a182;
int a3;	int a23;	int a43;	int a63;	int a83;	int a103;	int a123;	int a143;	int a163;	int a183;
int a4;	int a24;	int a44;	int a64;	int a84;	int a104;	int a124;	int a144;	int a164;	int a184;
int a5;	int a25;	int a45;	int a65;	int a85;	int a105;	int a125;	int a145;	int a165;	int a185;
int a6;	int a26;	int a46;	int a66;	int a86;	int a106;	int a126;	int a146;	int a166;	int a186;
int a7;	int a27;	int a47;	int a67;	int a87;	int a107;	int a127;	int a147;	int a167;	int a187;
int a8;	int a28;	int a48;	int a68;	int a88;	int a108;	int a128;	int a148;	int a168;	int a188;
int a9;	int a29;	int a49;	int a69;	int a89;	int a109;	int a129;	int a149;	int a169;	int a189;
int a10;	int a30;	int a50;	int a70;	int a90;	int a110;	int a130;	int a150;	int a170;	int a190;
int a11;	int a31;	int a51;	int a71;	int a91;	int a111;	int a131;	int a151;	int a171;	int a191;
int a12;	int a32;	int a52;	int a72;	int a92;	int a112;	int a132;	int a152;	int a172;	int a192;
int a13;	int a33;	int a53;	int a73;	int a93;	int a113;	int a133;	int a153;	int a173;	int a193;
int a14;	int a34;	int a54;	int a74;	int a94;	int a114;	int a134;	int a154;	int a174;	int a194;
int a15;	int a35;	int a55;	int a75;	int a95;	int a115;	int a135;	int a155;	int a175;	int a195;
int a16;	int a36;	int a56;	int a76;	int a96;	int a116;	int a136;	int a156;	int a176;	int a196;
int a17;	int a37;	int a57;	int a77;	int a97;	int a117;	int a137;	int a157;	int a177;	int a197;
int a18;	int a38;	int a58;	int a78;	int a98;	int a118;	int a138;	int a158;	int a178;	int a198;
int a19;	int a39;	int a59;	int a79;	int a99;	int a119;	int a139;	int a159;	int a179;	int a199;
int a20;	int a40;	int a60;	int a80;	int a100;	int a120;	int a140;	int a160;	int a180;	int a200;

所以我們需要

陣列

何謂陣列？

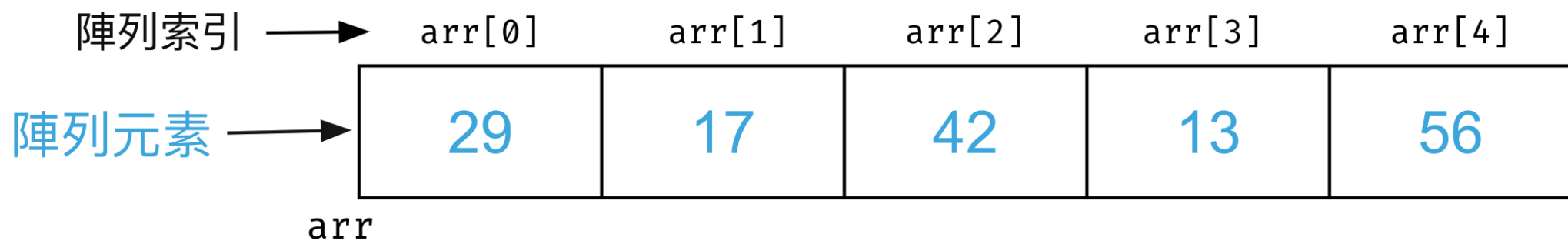
# 何謂陣列？

**陣列(array)** 是一種用來儲存資料的資料結構

陣列中的每個元素都是相同的資料型態

利用 **索引(index)** 就可以找出對應的元素(element)

## 長度為5的陣列



- 因此一個長度為5的陣列就有索引為0, 1, 2, 3, 4的五個元素
  - \*陣列的第一個元素是從0開始的

## 陣列的特性：

# 陣列的特性：

- **相同類型的元素**

所有元素都是相同類型的，例如整數、浮點數、字元等。

# 陣列的特性：

- **相同類型的元素**

所有元素都是相同類型的，例如整數、浮點數、字元等。

- **連續的記憶體空間**

陣列的元素在記憶體中是連續存放的。



# 陣列的特性：

- **相同類型的元素**

所有元素都是相同類型的，例如整數、浮點數、字元等。

- **連續的記憶體空間**

陣列的元素在記憶體中是連續存放的。

- **索引訪問**

每個元素都有一個唯一的索引，可以使用索引來訪問和修改特定位置的元素。

例如：`arr[3]`，`arr[0]`

\*索引值必  $> 0$

# 陣列的特性：

- **相同類型的元素**

所有元素都是相同類型的，例如整數、浮點數、字元等。

- **連續的記憶體空間**

陣列的元素在記憶體中是連續存放的。

- **索引訪問**

每個元素都有一個唯一的索引，可以使用索引來訪問和修改特定位置的元素。

例如：`arr[3]`，`arr[0]`

\*索引值必  $> 0$

- **固定大小**

陣列在創建時需要指定大小，且一旦分配了大小，通常不能動態改變。

# 陣列的宣告：

## ■ 創建一個整數(int)陣列

```
// 給定數值編譯器自動判斷陣列大小
```

```
int integerArray[] = {2,7,6,8,3};
```

```
// 創建一個長度5的int陣列，不給定初始值
```

```
int integerArray2[5];
```

```
// 創建一個長度5的int陣列，給定預設初始值(0)
```

```
int integerArray3[5]={};
```

```
// 創建一個長度5的int陣列，給定某些初始值，其它為0
```

```
int integerArray3[5]={2,3}; // 2,3,0,0,0
```

## ■ 創建一個浮點(float/double)陣列

```
float arr1[] = {2.3, 5.4, 9.3, 8.03};
```

```
double arr2[] = {8.3, 2.7, 3.6};
```

# Array 範例1

```
1  #include <stdio.h>
2  int main(){
3      int arr[5] = {2,4,7,8,3};
4      for (int i=0; i<5; i++){
5          printf("%d ", arr[i]);
6      }
7      printf("\n");
8      return 0;
9  }
```

# Array 範例1

```
1  #include <stdio.h>
2  int main(){
3      int arr[5] = {2,4,7,8,3};
4      for (int i=0; i<5; i++){
5          printf("%d ", arr[i]);
6      }
7      printf("\n");
8      return 0;
9  }
```

印出：

2 4 7 8 3

## Array 範例2.1

```
1  #include <stdio.h>
2  int main(){
3      int arr[4];
4      for (int i=0; i<4; i++){
5          scanf("%d", &arr[i]);
6      }
7
8      int sum = 0;
9      for (int i=0; i<4; i++){
10         sum += arr[i];
11     }
12
13     printf("sum = %d\n", sum);
14     printf("avg = %d\n", sum/4);
15     return 0;
16 }
```

輸入：

5 13 4 9

## Array 範例2.1

```
1  #include <stdio.h>
2  int main(){
3      int arr[4];
4      for (int i=0; i<4; i++){
5          scanf("%d", &arr[i]);
6      }
7
8      int sum = 0;
9      for (int i=0; i<4; i++){
10         sum += arr[i];
11     }
12
13     printf("sum = %d\n", sum);
14     printf("avg = %d\n", sum/4);
15     return 0;
16 }
```

輸入：

5 13 4 9

輸出：

sum = 31  
avg = 7

## Array 範例2.2

```
1  #include <stdio.h>
2  int main(){
3      int arr[4];
4      for (int i=0; i<4; i++){
5          scanf("%d", &arr[i]);
6      }
7
8      int sum = 0;
9      for (int i=0; i<4; i++){
10         sum += arr[i];
11     }
12
13     printf("sum = %d\n", sum);
14     printf("avg = %.2lf\n", (double)sum/4);
15     return 0;
16 }
```

輸入：

5 13 4 9



## Array 範例2.2

```
1  #include <stdio.h>
2  int main(){
3      int arr[4];
4      for (int i=0; i<4; i++){
5          scanf("%d", &arr[i]);
6      }
7
8      int sum = 0;
9      for (int i=0; i<4; i++){
10         sum += arr[i];
11     }
12
13     printf("sum = %d\n", sum);
14     printf("avg = %.2lf\n", (double)sum/4);
15     return 0;
16 }
```

輸入：

5 13 4 9

輸出：

sum = 31  
avg = 7.75

字符串

# 字串是什麼？

其實字串就是一種字元陣列

# 字串的宣告

```
char s[] = {'a', 'p', 'p', 'l', 'e', '\0'};
```

``\0`` 是結束字元，代表字串結尾。

```
char s[] = "apple";
```

與上面相同效果

## 字串範例1：

```
1  #include <stdio.h>
2  int main(){
3      char s1[] = "Hello, World";
4      char s2[] = {'H', 'E', 'L', 'L', 'O', '\0'};
5
6      printf("%c %c\n", s1[4], s2[1]);
7      printf("%s\n", s1);
8      printf("%s\n", s2);
9
10     return 0;
11 }
```

## 字串範例1：

```
1  #include <stdio.h>
2  int main(){
3      char s1[] = "Hello, World";
4      char s2[] = {'H', 'E', 'L', 'L', 'O', '\0'};
5
6      printf("%c %c\n", s1[4], s2[1]);
7      printf("%s\n", s1);
8      printf("%s\n", s2);
9
10     return 0;
11 }
```

輸出：

```
o E
Hello, World
HELLO
```

## 字串範例2：

```
1  #include <stdio.h>
2  int main(){
3      char myStr[20];
4      scanf("%s", myStr);
5      printf("Hi, %s\n", myStr);
6      return 0;
7  }
```

輸入：

C\_String

## 字串範例2：

```
1  #include <stdio.h>
2  int main(){
3      char myStr[20];
4      scanf("%s", myStr);
5      printf("Hi, %s\n", myStr);
6      return 0;
7  }
```

輸入：

C\_String

index	0	1	2	3	4	5	6	7	8	9	10	...	19
內容	'C'	'_'	'S'	't'	'r'	'i'	'n'	'g'	'\0'	undefined	undefined	...	undefined



## 字串範例2：

```
1  #include <stdio.h>
2  int main(){
3      char myStr[20];
4      scanf("%s", myStr);
5      printf("Hi, %s\n", myStr);
6      return 0;
7  }
```

輸入：

C\_String

輸出：

Hi, C\_String

index	0	1	2	3	4	5	6	7	8	9		10	...	19
內容	'C'	'_'	'S'	't'	'r'	'i'	'n'	'g'	'\0'	undefined		undefined	...	undefined

## 補充：其它字串相關函式

```
// 使用以下函式需要引入此標頭檔
```

```
#include <string.h>
```

```
// 將一字串之內容複製到另一字串
```

```
strcpy(destination, source);
```

```
// 比較字串之字典序
```

```
strcmp(s1, s2)
```

```
// 將s2的內容加到s1之後
```

```
strcat(s1, s2);
```

# 6. 指標 Pointers

(使用C)

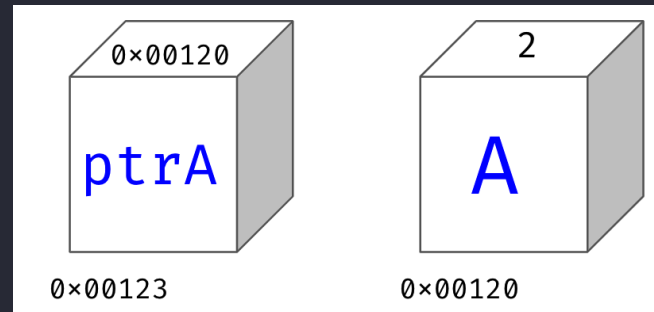
作者：劉宸均

# 何謂指標？

一種特殊的變數，它存儲的是一個記憶體位址

```
1  #include <stdio.h>
2  int main(){
3      int A = 3;
4      int *ptrA;
5      ptrA = &A; // A在記憶體中的位址
6  }
```

`&`：取址運算子



由於指標本身也是資料的一種，因此他也具有一般資料所有的資訊：

- 名稱：`ptrA`
- 資料型態：`int *`
- 資料內容：`0x00120`
- `ptrA`本身記憶體位置：`0x00123`

# 指標範例1

```
1  #include <stdio.h>
2  int main(){
3      int A = 3;
4      int *ptrA = &A;
5
6      printf("%d\n", *ptrA);
7      *ptrA = 50;
8      printf("%d\n", A);
9  }
```

`\*` 取值運算子

# 指標範例1

```
1  #include <stdio.h>
2  int main(){
3      int A = 3;
4      int *ptrA = &A;
5
6      printf("%d\n", *ptrA);
7      *ptrA = 50;
8      printf("%d\n", A);
9  }
```

`\*ptrA` 就是 A

`\*` 取值運算子

# 指標範例1

```
1  #include <stdio.h>
2  int main(){
3      int A = 3;
4      int *ptrA = &A;
5
6      printf("%d\n", *ptrA);
7      *ptrA = 50;
8      printf("%d\n", A);
9  }
```

`\*ptrA` 就是 A

`\*ptrA = 50`

透過(\*ptrA)去更動A的  
值。

`\*` 取值運算子

# 指標範例1

```
1  #include <stdio.h>
2  int main(){
3      int A = 3;
4      int *ptrA = &A;
5
6      printf("%d\n", *ptrA);
7      *ptrA = 50;
8      printf("%d\n", A);
9  }
```

`\*ptrA` 就是 A

`\*ptrA = 50`

透過(\*ptrA)去更動A的  
值。

`\*` 取值運算子



# 指標範例1

```
1  #include <stdio.h>
2  int main(){
3      int A = 3;
4      int *ptrA = &A;
5
6      printf("%d\n", *ptrA);
7      *ptrA = 50;
8      printf("%d\n", A);
9  }
```

`\*` 取值運算子

`\*ptrA` 就是 A

`\*ptrA = 50`

透過(\*ptrA)去更動A的  
值。

輸出：

3  
50

# 陣列的運作原理

```
int arr[10];
```

`arr` 是一個指標，指向陣列頭所在位址

`arr` 是 `arr[0]` 的所在位址

`\*arr` 就是 `arr[0]`

`arr+1` 是 `arr[1]` 的所在位址

`\*(arr+1)` 就是 `arr[1]`

`arr+2` 是 `arr[2]` 的所在位址

`\*(arr+2)` 就是 `arr[2]`

# 指標與陣列 範例1

```
1  #include <stdio.h>
2  int main(){
3      int arr[5];
4      for (int i=0; i<5; i++)
5          scanf("%d", arr+i); // 和 &arr[i] 一樣
6      for (int i=0; i<5; i++)
7          printf("%d ", *(arr+i)); // 和 arr[i] 一樣
8      printf("\n");
9      return 0;
10 }
```

輸入：

3 7 2 9 4

# 指標與陣列 範例1

```
1  #include <stdio.h>
2  int main(){
3      int arr[5];
4      for (int i=0; i<5; i++)
5          scanf("%d", arr+i); // 和 &arr[i] 一樣
6      for (int i=0; i<5; i++)
7          printf("%d ", *(arr+i)); // 和 arr[i] 一樣
8      printf("\n");
9      return 0;
10 }
```

輸入：

3 7 2 9 4

# 指標與陣列 範例1

```
1  #include <stdio.h>
2  int main(){
3      int arr[5];
4      for (int i=0; i<5; i++)
5          scanf("%d", arr+i); // 和 &arr[i] 一樣
6      for (int i=0; i<5; i++)
7          printf("%d ", *(arr+i)); // 和 arr[i] 一樣
8      printf("\n");
9      return 0;
10 }
```

輸入：

3 7 2 9 4

# 指標與陣列 範例1

```
1  #include <stdio.h>
2  int main(){
3      int arr[5];
4      for (int i=0; i<5; i++)
5          scanf("%d", arr+i); // 和 &arr[i] 一樣
6      for (int i=0; i<5; i++)
7          printf("%d ", *(arr+i)); // 和 arr[i] 一樣
8      printf("\n");
9      return 0;
10 }
```

輸入：

3 7 2 9 4

# 指標與陣列 範例1

```
1  #include <stdio.h>
2  int main(){
3      int arr[5];
4      for (int i=0; i<5; i++)
5          scanf("%d", arr+i); // 和 &arr[i] 一樣
6      for (int i=0; i<5; i++)
7          printf("%d ", *(arr+i)); // 和 arr[i] 一樣
8      printf("\n");
9      return 0;
10 }
```

輸入：

3 7 2 9 4

輸出：

3 7 2 9 4

## 指標與陣列 範例2

```
1  #include <stdio.h>
2  int main(){
3      int arr[5] = {9, 3, 1, 0, 8};
4      printf("%d\n", *(arr+1));
5      printf("%d\n", *arr+1);
6
7      return 0;
8  }
```



## 指標與陣列 範例2

```
1  #include <stdio.h>
2  int main(){
3      int arr[5] = {9, 3, 1, 0, 8};
4      printf("%d\n", *(arr+1));
5      printf("%d\n", *arr+1);
6
7      return 0;
8  }
```

輸出：

```
3
10
```

# 二維陣列

```
int arr[3][4]; // 3x4 2d array. (3 rows, 4 columns)
```

row      column

4 columns

3 rows

	column 0	column 1	column 2	column 3
row 0	arr[0][0]	arr[0][1]	arr[0][2]	arr[0][3]
row 1	arr[1][0]	arr[1][1]	arr[1][2]	arr[1][3]
row 2	arr[2][0]	arr[2][1]	arr[2][2]	arr[2][3]

# 二維陣列

```
int mtx[2][3] = { {1, 4, 2}, {3, 6, 8} };
```

	COLUMN 0	COLUMN 1	COLUMN 2
ROW 0	1	4	2
ROW 1	3	6	8

在記憶體中的排列方式：

mtx[0][0]	mtx[0][1]	mtx[0][2]	mtx[1][0]	mtx[1][1]	mtx[1][2]
1	4	2	3	6	8
mtx	mtx+1	mtx+2	mtx+3	mtx+4	mtx+5