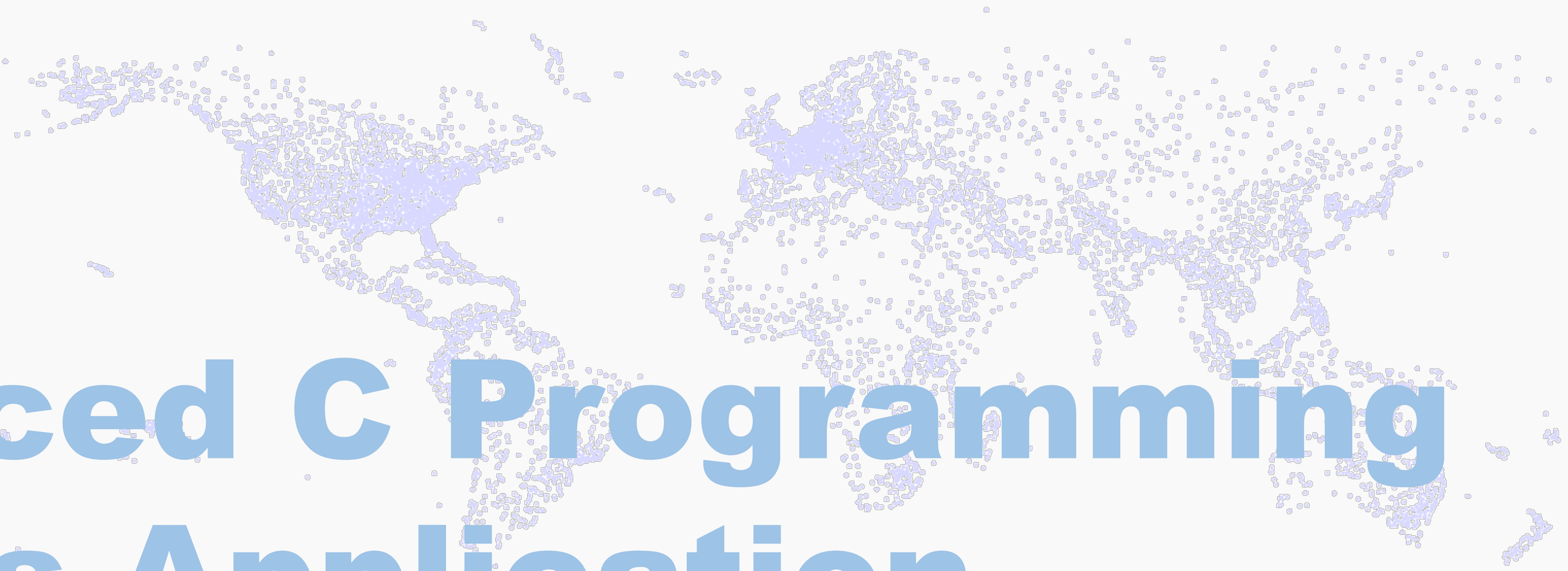


Advanced C Programming And It's Application

Pointer III: Array of Pointers & Pointer to 2D Array

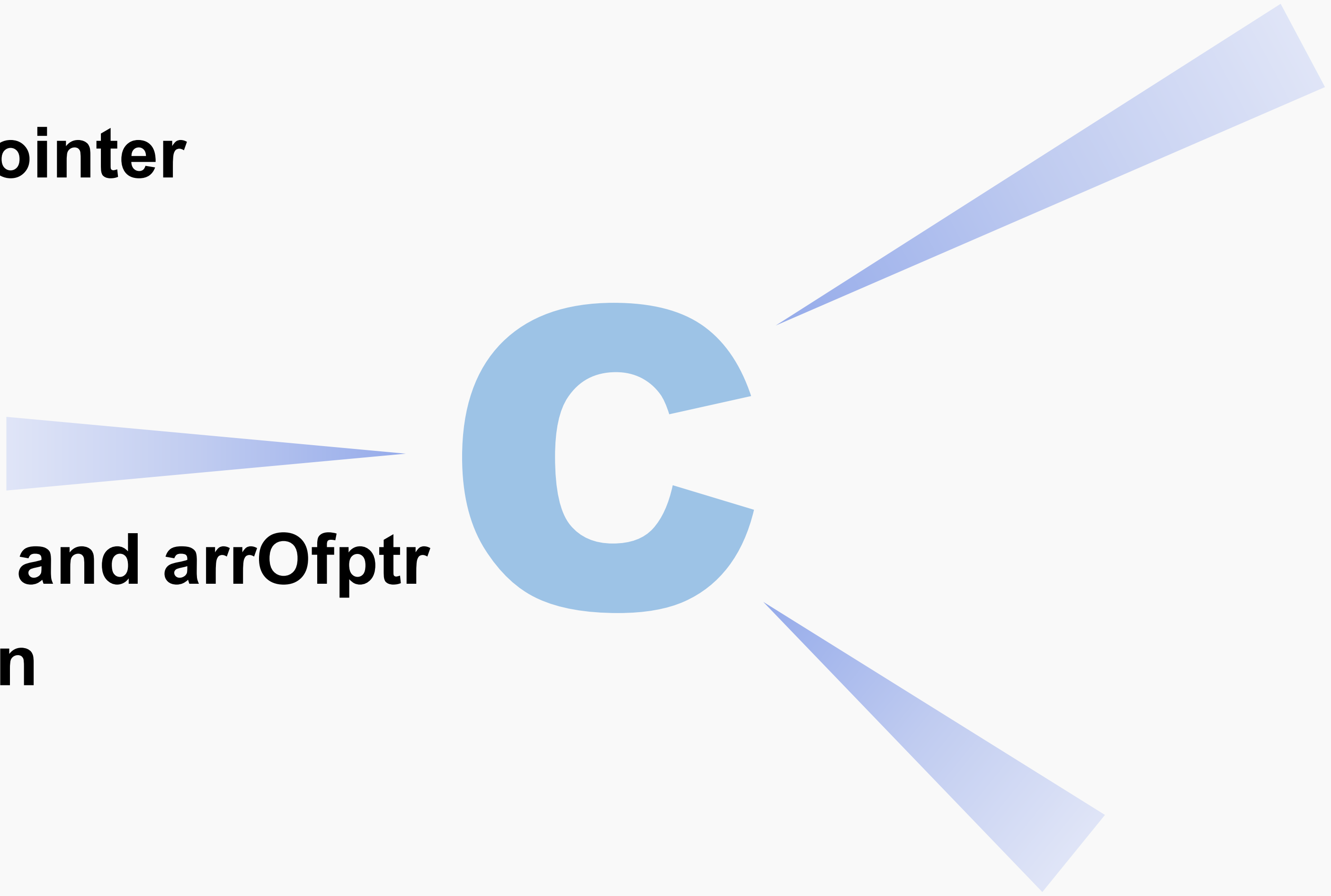
Assistant Prof. Chan, Chun-Hsiang

Department of Artificial Intelligence, Tamkang University

Nov. 10, 2021

<Outline/>

大綱



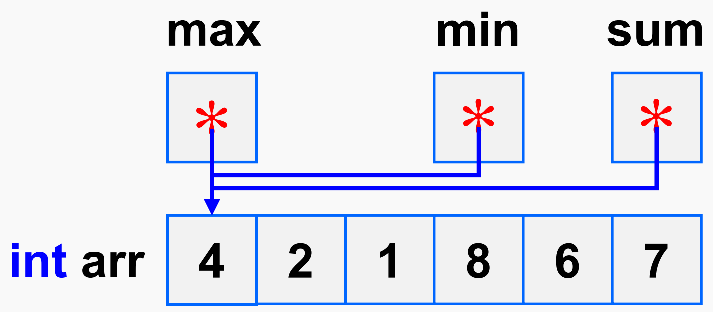
</Outline>

Pointer to Pointer

「Pointer to Pointer」到底有甚麼應用呢?

如果我們要有一些列出矩陣的屬性資料(e.g., size, max, min, & sum)，我們要如何將這些屬性與這個矩陣做連結? 同時我們又希望可以把找屬性的這件事情，變成一個函數，每次使用時都直接進來呼叫，並得到我們要的結果。

這個時候「Pointer to Pointer」就很好用了!



Pointer to Pointer

Find max value – return value

```
#include <stdio.h>
int myMax(int a[], int s){
	int max = 0, i;
	for (i=0;i<s;i++){
		if (a[i]>max){
			max = a[i];
		}
	}
	return max;
}
```

```
int main(){
	/*Ex 7-1: ptr2ptr :: find max in an array*/
	printf("Ex 7-1: ptr2ptr :: find max in an array\n");
	int arr[5] = {51,41,311,211,110};
	int size = 5;
	printf("%d\n",myMax(arr, size));
}
```

在不會pointer之前，如果要你尋找一個矩陣的最大值，你的找法應該是會這樣!

Pointer to Pointer

Find max value – return a pointer

```
#include <stdio.h>
int *my_max(int a[], int s){
        int *max = a, i;
        for (i=0;i<s;i++){
                if (a[i]>*max){
                        max = &a[i];
                }
        }
        return max;
}
```

```
int main(){
        /*Ex 7-2: ptr2ptr :: find max in an array*/
        printf("Ex 7-2: ptr2ptr :: find max in an array\n");
        int arr[5] = {51,41,311,211,110};
        int size = 5;
        printf("%d\n",*my_max(arr,size));
}
```

學過pointer之後你可以直接回傳pointer回來!

Pointer to Pointer

Find max value – pointer to pointer

```
#include <stdio.h>
void my_max(int a[], int s, int **ptrmax){
        *ptrmax = a;
        int i;
        for (i=0;i<s;i++){
                if (a[i]>**ptrmax){
                        *ptrmax = &a[i];
                }
        }
}
```

```
int main(){
        /*Ex 7-3: ptr2ptr :: find max in an
        array*/
        printf("Ex 7-3: ptr2ptr :: find max in
        an array\n");
        int arr[5] = {51,41,311,211,110};
        int size = 5, *max;
        my_max(arr, size, &max);
        printf("max: %d\n",*max);
}
```

直接call矩陣回傳的是一個地址 ➔ 矩陣也是個pointer!
所以說可以用pointer to pointer指向一個矩陣變數arr!

Pointer to Pointer

Lab 7-1:

請利用Ex 7-1 ~ 7-3的方式，實做出尋找最小值以及回傳最小值的位置(矩陣中的位置)。

測試資料:

int arr[10] = {5,7,9,3,4,0,6,1,2,8};

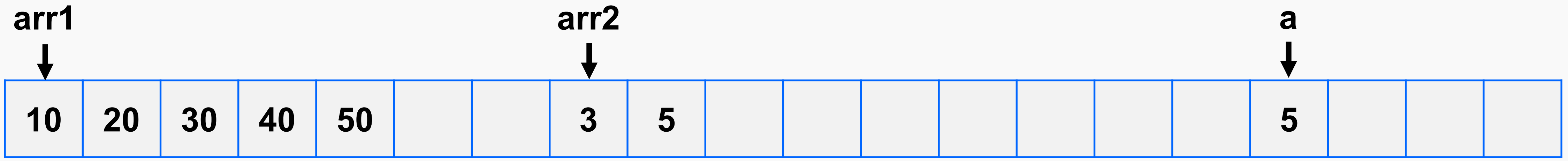
測試結果:

最小值為**0**，位置在第**5**個元素 (從0開始算)。

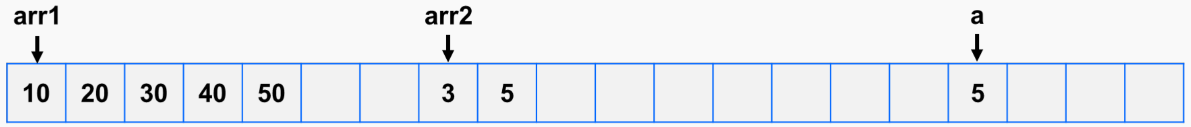
Array of Pointers

前面我們討論過如何利用指標來存取矩陣的屬性或一些特徵，那如果今天我想要將矩陣合併，或是將同資料型別的東西串起來，那應該如何去做呢?

```
int arr1[5] = {10,20,30,40,50};
int arr2[2] = {3,5};
int a = 5;
```



Array of Pointers



```
#include <stdio.h>
int main(){
        /*Ex 7-4: array of pointers*/
        printf("Ex 7-4: array of pointers\n");
        int arr1[5] = {10, 20, 30, 40, 50};
        int arr2[2] = {3,5};
        int a = 5, i;
        int *arrOfPtr[4] = {arr1, &a, arr1+3};
        arrOfPtr[3] = arr2;
        for (i=0; i<4; i++){
                printf("%d\t%p\n", *arrOfPtr[i], arrOfPtr[i]);
        }
}
```

Array of Pointers

```
#include <stdio.h>
int main(){
        /*Ex 7-5: array of pointers: print */
        printf("Ex 7-5: array of pointers: print\n");
        int arr1[10] = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100}, arr2[5] = {11,12,13,14,15};
        int arr3[3] = {111,112,113}, i, j;
        int *arrOfPtr[3] = {arr1, arr2, arr3};
        int s[3] = {10,5,3};
        for (i=0; i<3; i++){
                for (j=0; j<s[i]; j++){
                        printf("%d\t", arrOfPtr[i][j]);
                }
                printf("\n");
        }
}
```

Pointer to Array

在array of pointers之後，我們就是要來看到pointer to array，用pointer指向array。在開始之前，為了讓大家喚起記憶，我們先做以下簡短的小練習:

```
#include <stdio.h>
int main(){
        int arr[4];
        int *p = arr;
        printf("p    is %p, while p+1    is %p\n", p, p+1);
        printf("&arr is %p, while &arr+1 is %p\n", &arr, &arr+1);
}
```

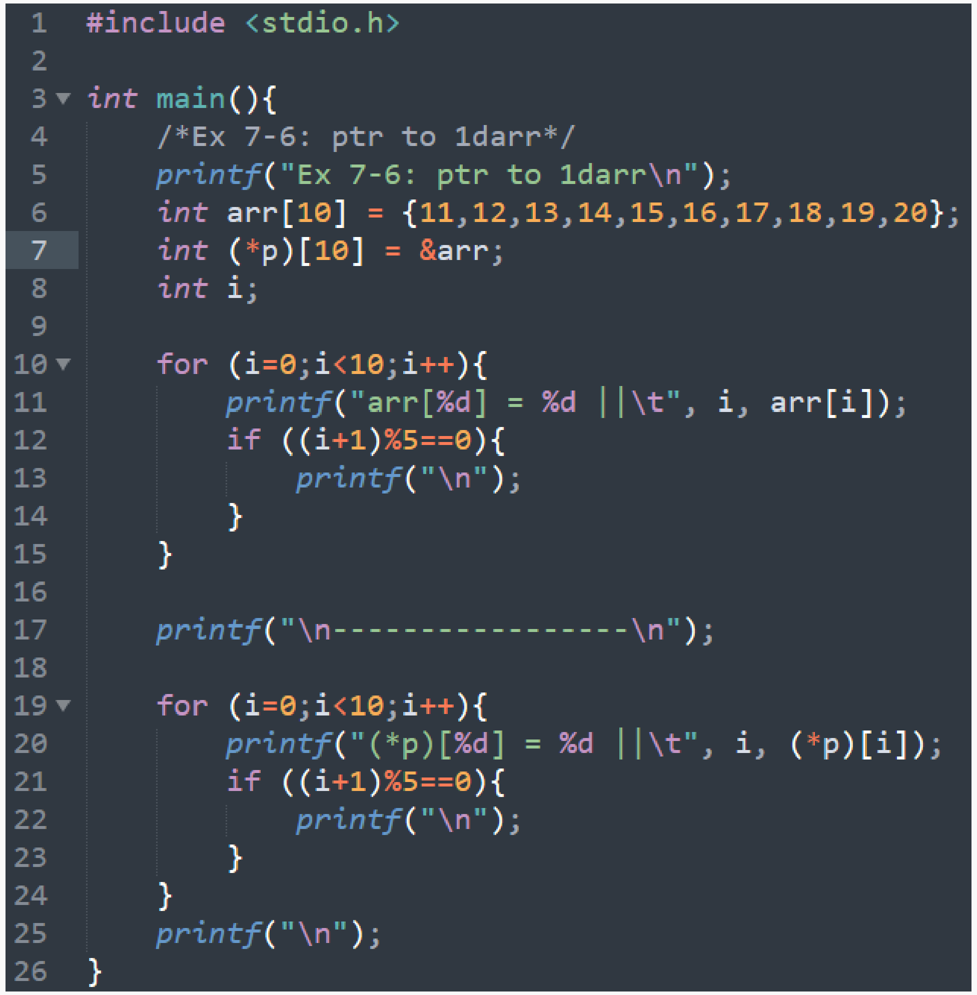
Lab 7-2:

思考這下列兩題的物理意義:

(1) (&p+1)-&p

(2) (&arr+1) - &arr

Pointer to Array



```c
#include <stdio.h>

int main(){
    /*Ex 7-6: ptr to 1darr*/
    printf("Ex 7-6: ptr to 1darr\n");
    int arr[10] = {11,12,13,14,15,16,17,18,19,20};
    int (*p)[10] = &arr;
    int i;

    for (i=0;i<10;i++){
        printf("arr[%d] = %d ||\t", i, arr[i]);
        if ((i+1)%5==0){
            printf("\n");
        }
    }

    printf("\n----------------\n");

    for (i=0;i<10;i++){
        printf("(*p)[%d] = %d ||\t", i, (*p)[i]);
        if ((i+1)%5==0){
            printf("\n");
        }
    }
    printf("\n");
}
```

11	12	13	14	15	16	17	18	19	20

Note:

arr[i] ➔ int *

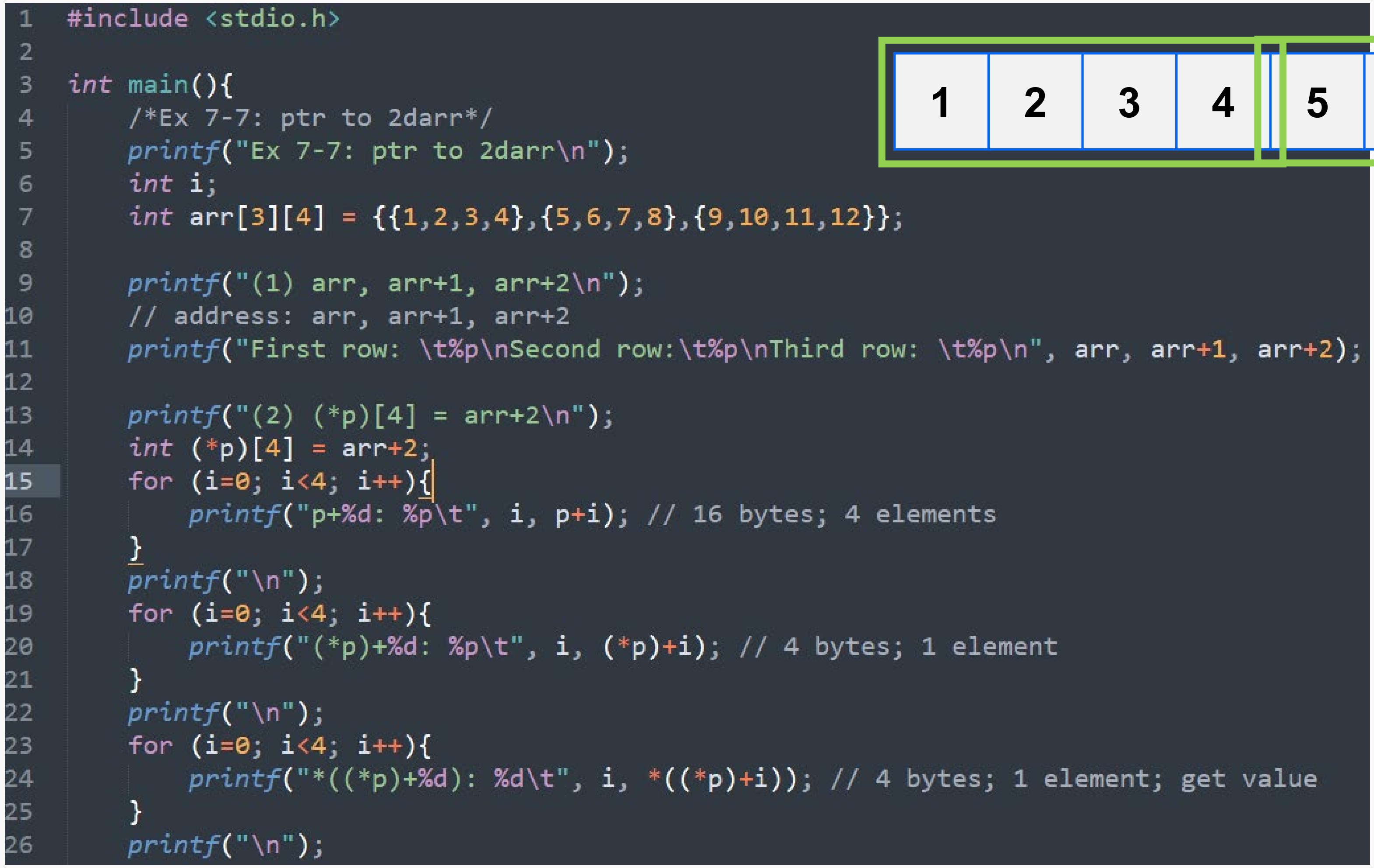
∴ an array is consisted of several pointers s.t. we can use (*p)[10] to represent arr.

∵ an array in C has two characteristics:

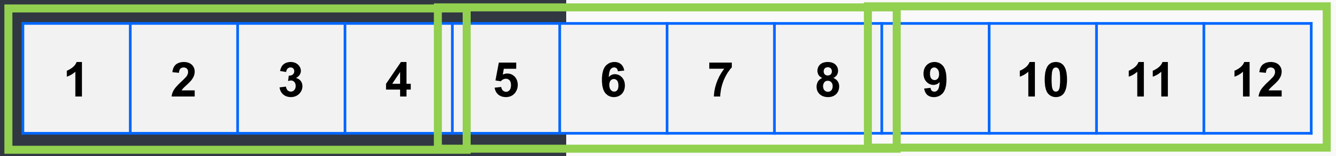
1. **“Array of pointers” to store all element locations**
2. **“Pointer to array” to indicate the first element of the array**

</ ptr to arr>

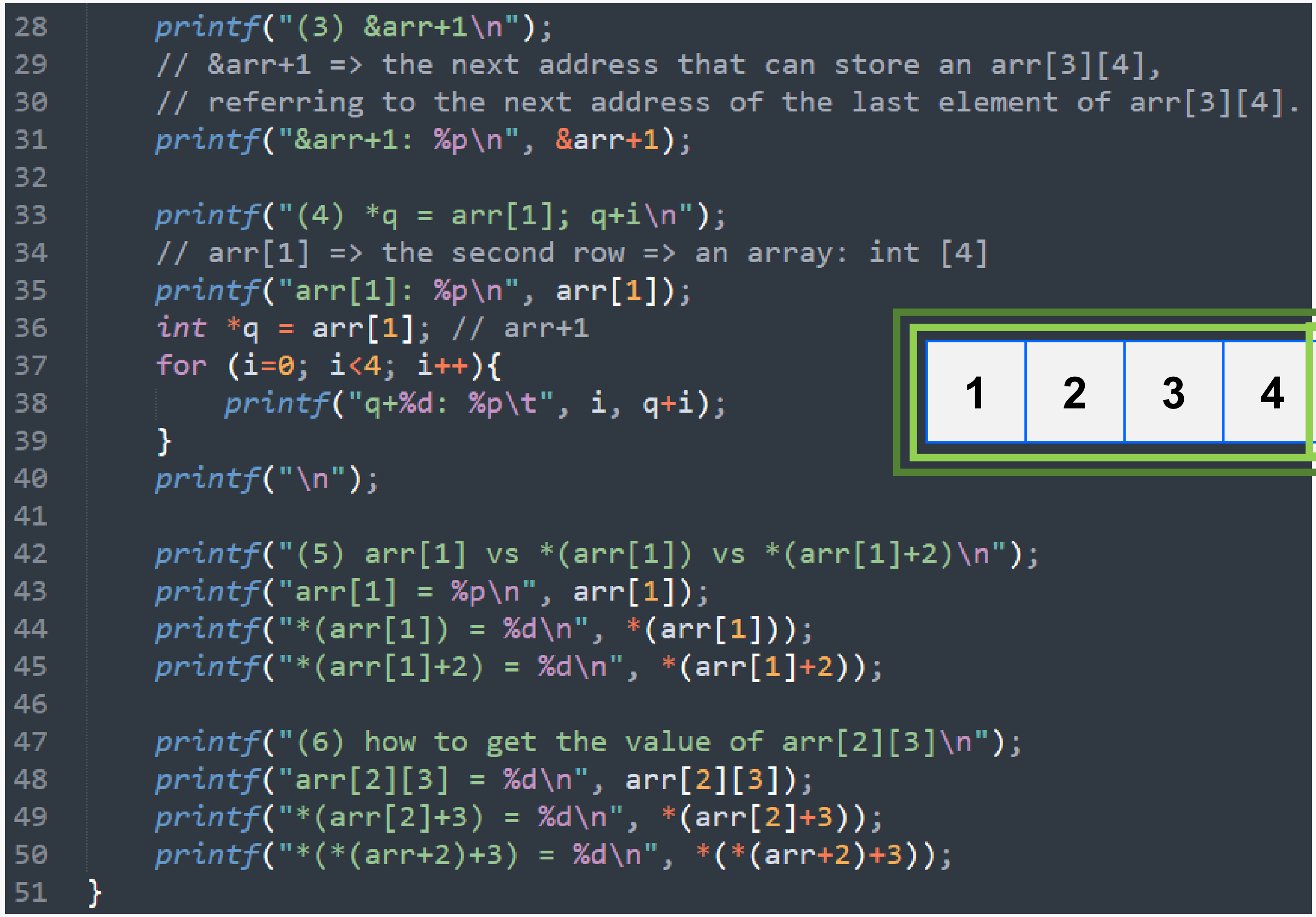
Pointer to Array



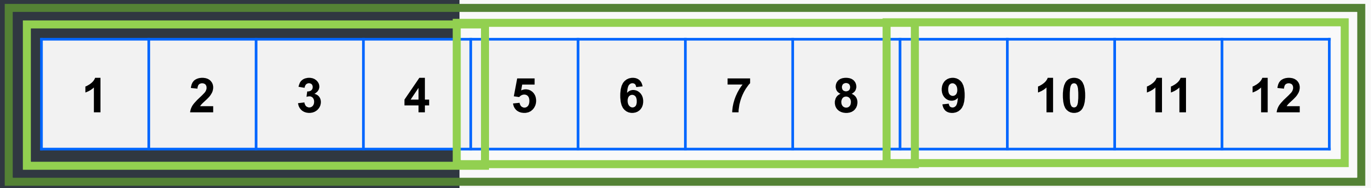
```c
#include <stdio.h>

int main(){
    /*Ex 7-7: ptr to 2darr*/
    printf("Ex 7-7: ptr to 2darr\n");
    int i;
    int arr[3][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};

    printf("(1) arr, arr+1, arr+2\n");
    // address: arr, arr+1, arr+2
    printf("First row: \t%p\nSecond row:\t%p\nThird row: \t%p\n", arr, arr+1, arr+2);

    printf("(2) (*p)[4] = arr+2\n");
    int (*p)[4] = arr+2;
    for (i=0; i<4; i++){
        printf("p+%d: %p\t", i, p+i); // 16 bytes; 4 elements
    }
    printf("\n");
    for (i=0; i<4; i++){
        printf("(*p)+%d: %p\t", i, (*p)+i); // 4 bytes; 1 element
    }
    printf("\n");
    for (i=0; i<4; i++){
        printf("*((*p)+%d): %d\t", i, *((*p)+i)); // 4 bytes; 1 element; get value
    }
    printf("\n");
```



Pointer to Array



```
    printf("(3) &arr+1\n");
    // &arr+1 => the next address that can store an arr[3][4],
    // referring to the next address of the last element of arr[3][4].
    printf("&arr+1: %p\n", &arr+1);

    printf("(4) *q = arr[1]; q+i\n");
    // arr[1] => the second row => an array: int [4]
    printf("arr[1]: %p\n", arr[1]);
    int *q = arr[1]; // arr+1
    for (i=0; i<4; i++){
        printf("q+%d: %p\t", i, q+i);
    }
    printf("\n");

    printf("(5) arr[1] vs *(arr[1]) vs *(arr[1]+2)\n");
    printf("arr[1] = %p\n", arr[1]);
    printf("*(arr[1]) = %d\n", *(arr[1]));
    printf("*(arr[1]+2) = %d\n", *(arr[1]+2));

    printf("(6) how to get the value of arr[2][3]\n");
    printf("arr[2][3] = %d\n", arr[2][3]);
    printf("*(arr[2]+3) = %d\n", *(arr[2]+3));
    printf("*(*(arr+2)+3) = %d\n", *(*(arr+2)+3));
}
```



Pointer to Array

前面提過array擁有兩個特性:

(1) Pointer to array

(2) Array of pointers

也因為如此，不同表示式代表著不同的意義...

以Ex 7-7 為例，int arr[3][4] =

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

Expression	Type
arr	???
arr+0	???
arr+2	???
arr[1]	???
arr[1]+1	???
arr[2][3]	???

Lab 7-3:

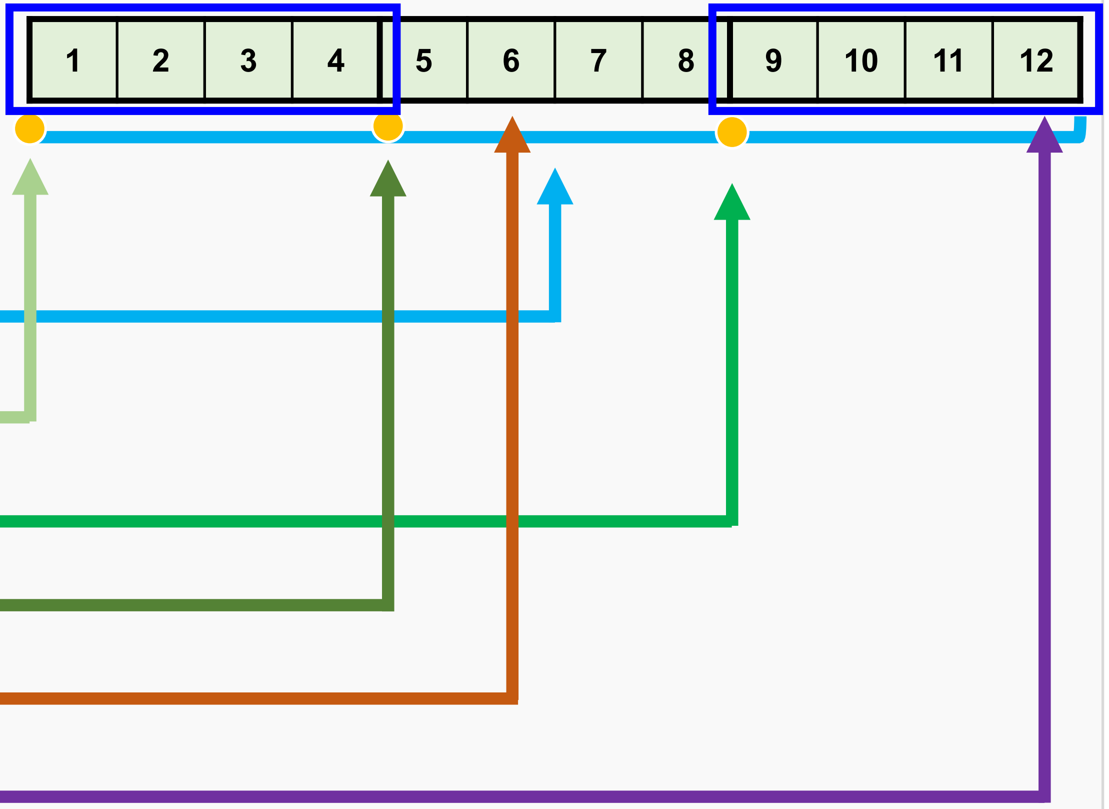
思考一下! *(arr[1]+3) ≡ *(*(arr+1)+3)?

Pointer to Array

以Ex 7-7 為例，int arr[3][4] =

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

=



Expression	Type
arr	int [3][4]
arr+0	int (*)[4]
arr+2	int (*)[4]
arr[1]	int [4]
arr[1]+1	int *
arr[2][3]	int

Pointer to Array

Lab 7-4:

依照上一頁的做法，完成下表:

Expression	Type	Explanation
(1) arr +1		
(2) *(arr + 1)		
(3) arr[2]		
(4) arr[2]+1		
(5) *(arr[2]+1)		
(6) *(*(arr+1)+1)		
(7) arr[1]+0		

int arr[3][4] =

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

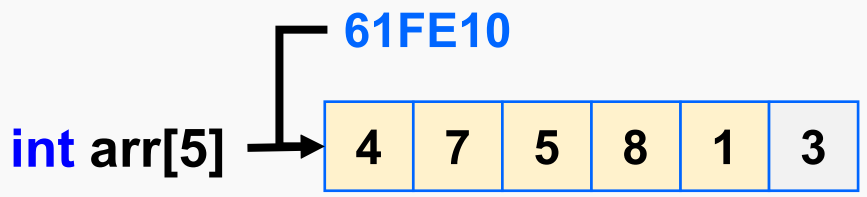
II

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

Length of Array

我們知道矩陣也是一個指標的時候，假設我今天對某個矩陣做了很多運算，然而我忘記剛剛在哪裡宣告這個矩陣，但是我想知道它的長度時該怎麼做?

```
int arr[5] = {4,7,5,8,1};
```



設計一個方法可以計算出來arr的長度!

Lab 7-5:

思考下列程式碼的回傳值?

(1) arr

(2) arr[1]

(3) arr+1

(4) &arr

(5) &arr + 1

Length of Array

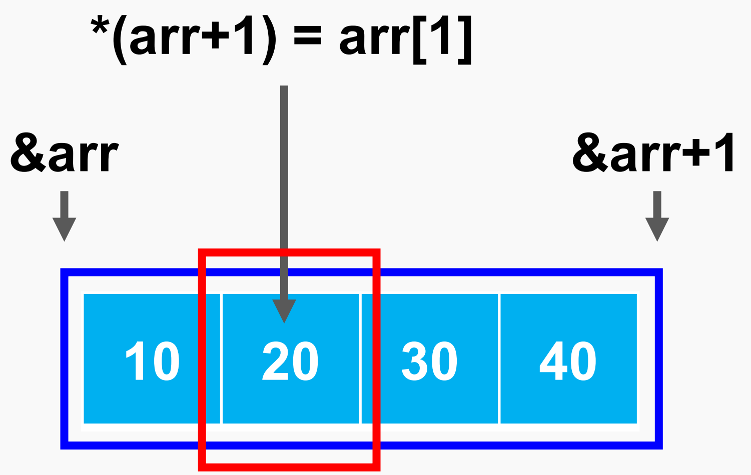
作法一: 利用內建函數sizeof()

```
#include <stdio.h>
int main(){
        /*Ex 7-8: length of array :: sizeof()*/
        printf("Ex 7-8: length of array :: sizeof()\n");
        int arr[4] = {10, 20, 30, 40};

        printf(" length = %d\n", sizeof(arr)/sizeof(arr[0]));
}
```

Length of Array

作法二: 利用矩陣的記憶體位置

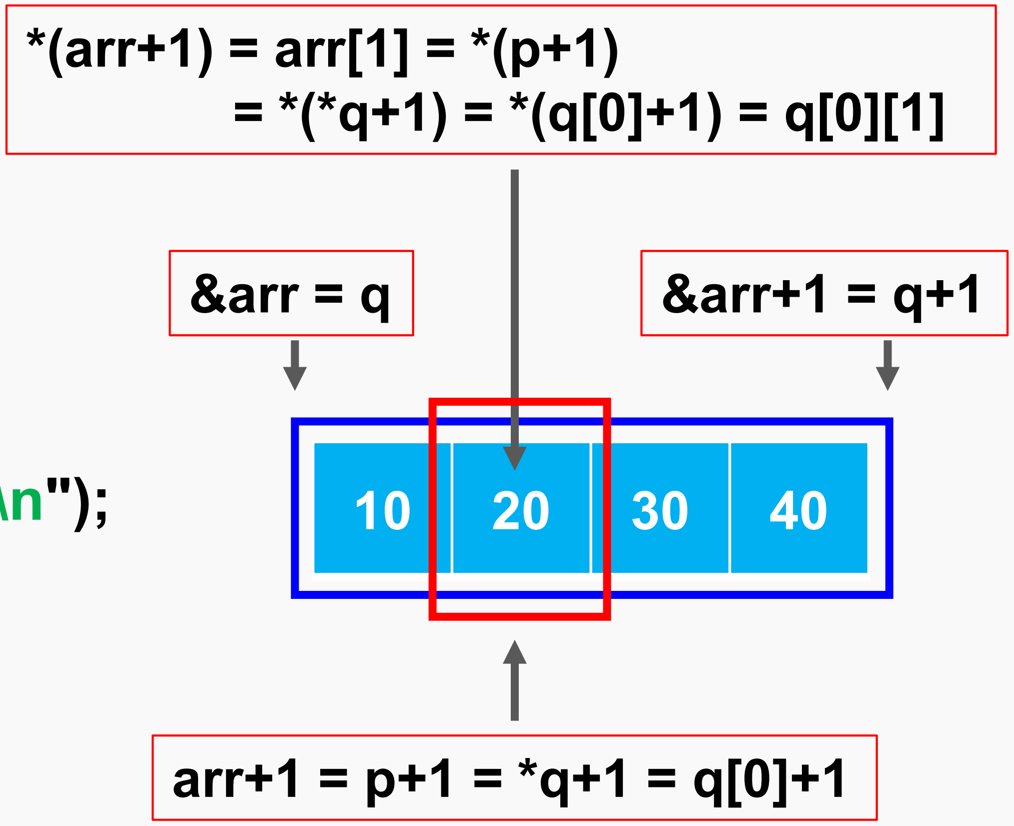
```
#include <stdio.h>
int main(){
        /*Ex 7-9: length of array :: address*/
        printf("Ex 7-9: length of array :: address\n");
        int arr[4] = {10, 20, 30, 40};
        printf("arr:       %p\n", arr); // int (*)[4]
        printf("arr+1:     %p\n", arr+1); // int *
        printf("&arr+1:    %p\n", &arr+1); // int (*)[4]
        printf("*(&arr+1): %p\n", *(&arr+1)); // int *
        printf("length = *(&arr+1)-arr: %d\n", *(&arr+1)-arr);
}
```



Length of Array

作法三: 利用矩陣的指標

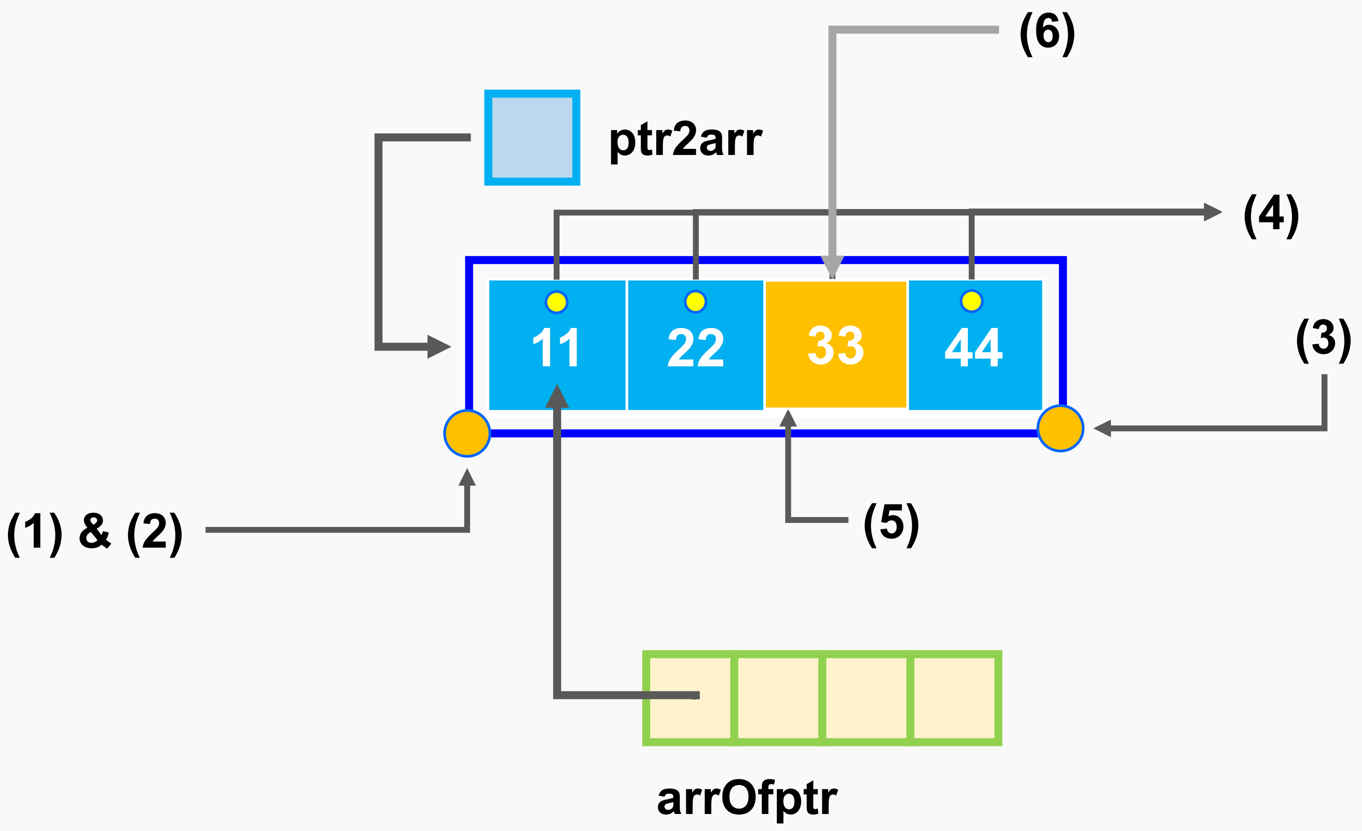
```
#include <stdio.h>
int main(){
        /*Ex 7-10: length of array :: pointer*/
        printf("Ex 7-10: length of array :: pointer\n");
        int arr[4] = {10, 20, 30, 40};
        int *p = arr;
        printf("p  : %p\n", p); // int *
        printf("p+1: %p\n", p+1); // int *
        int (*q)[4] = &arr;
        printf("q  : %p\n", *q); // int *
        printf("q+1: %p\n", *(q+1)); // int *
        printf("length = *(q+1)-*(q+0): %d\n", *(q+1)-*(q+0));
}
```


*(arr+1) = arr[1] = *(p+1)
= *(*q+1) = *(q[0]+1) = q[0][1]
&arr = q
&arr+1 = q+1
10
20
30
40
arr+1 = p+1 = *q+1 = q[0]+1

Summary of ptr2arr and arrOfptr

```
int arr[4] = {11, 22, 33, 44};
int (*ptr2arr)[4] = &arr;
int *arrOfptr[4];
arrOfptr[0] = arr;
```

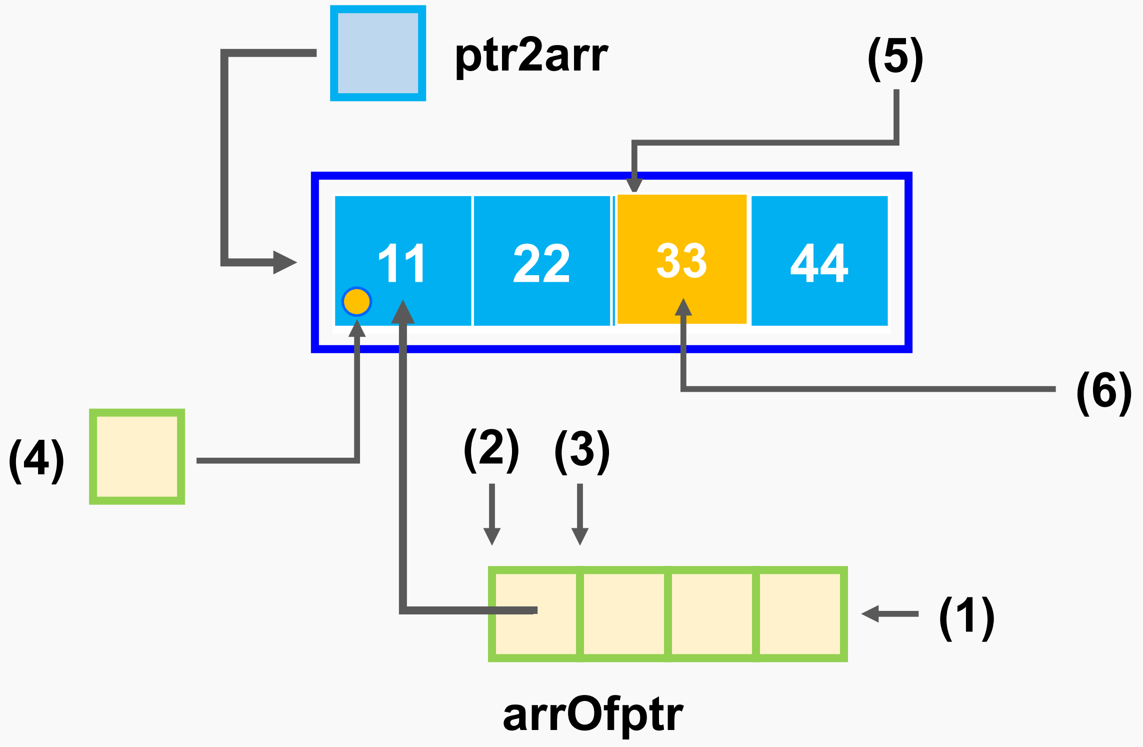
Expression	Type
(1) ptr2arr	int (*)[4]
(2) ptr2arr+0	int (*)[4]
(3) ptr2arr+1	int (*)[4]
(4) ptr2arr[0]	int [4]
(5) ptr2arr[0]+2	int *
(6) ptr2arr[0][2]	int



Summary of ptr2arr and arrOfptr

```
int arr[4] = {11, 22, 33, 44};
int (*ptr2arr)[4] = &arr;
int *arrOfptr[4];
arrOfptr[0] = arr;
```

Expression	Type
(1) arrOfptr	int *[4]
(2) arrOfptr+0	int **
(3) arrOfptr+1	int **
(4) arrOfptr[0]	int *
(5) arrOfptr[0]+2	int *
(6) arrOfptr[0][2]	int



2D array passing into Functions

在我們目前學到靜態宣告矩陣的情況下，一但我們知道矩陣維度、大小、還有第0個元素的位置，其實我們就可以很容易在其他函數中，找到它並針對它做運算。

```
#include <stdio.h>
void print2Darr(int (*p)[4], int row){
        // int (*p)[4] => change to other expressions...
        int i, j;
        for (i=0;i<row;i++){
                for (j=0;j<4;j++){
                        printf("%d\t", p[i][j]); // change here!
                }
                printf("\n");
        }
}

int main(){
        /*Ex 7-13: 2D arr passing to func*/
        printf("Ex 7-13: 2D arr passing to func\n");
        int arr[3][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12}};
        print2Darr(arr, 3);
}
```

Lab 7-6:

嘗試將Ex 7-13的兩處修改成其它的表示方式(每處至少寫兩種)，但不會改變其輸出的結果!

作業一

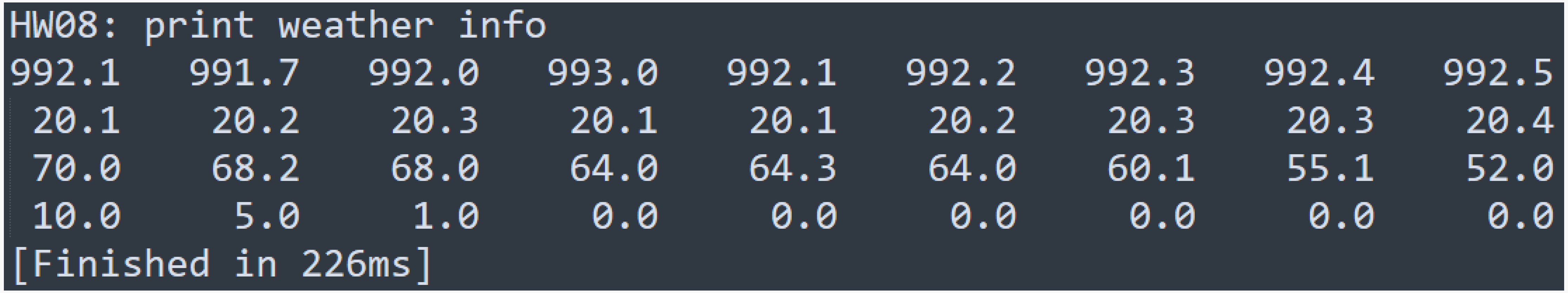
假設我們今天有一個氣象資料，它每小時會有一筆紀錄陽明山測站的氣壓、氣溫、濕度、降雨量資料，如下表所示:

Time	1	2	3	4	5	6	7	8	9
Pressure	992.1	991.7	992.0	993.0	992.1	992.2	992.3	992.4	992.5
Temperature	20.1	20.2	20.3	20.1	20.1	20.2	20.3	20.3	20.4
Humidity	70.0	68.2	68.0	64.0	64.3	64.0	60.1	55.1	52.0
Rainfall	10.0	5.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0

請將此資料利用一個函數(printWeatherInfo(int arg1, int arg2))將氣象紀錄印出，且arg1以及arg2其中一個必須為指標。

作業一

範例解答:



```
HW08: print weather info
992.1   991.7   992.0   993.0   992.1   992.2   992.3   992.4   992.5
 20.1    20.2    20.3    20.1    20.1    20.2    20.3    20.3    20.4
 70.0    68.2    68.0    64.0    64.3    64.0    60.1    55.1    52.0
 10.0     5.0     1.0     0.0     0.0     0.0     0.0     0.0     0.0
[Finished in 226ms]
```

References

- How to Build an Array of Pointers in C Programming: https://www.dummies.com/programming/c/how-to-build-an-array-of-pointers-in-c-programming/
- Array of Pointers in C: https://overiq.com/c-programming-101/array-of-pointers-in-c/
- Pointer to an Array in C https://www.tutorialspoint.com/cprogramming/c_pointer_to_an_array.htm
- (C) 簡單搞懂指標(pointer)、指標陣列(pointers of array, int *foo[]) 與指向陣列的指標 (pointer to array, int (*bar)[]) http://hackgrass.blogspot.com/2018/03/c-pointerint-foo-int-bar.html
- 蔣宗哲教授 – 程式設計(一) 講義