7. 指標
Pointers

作者：劉宸均

一般變數

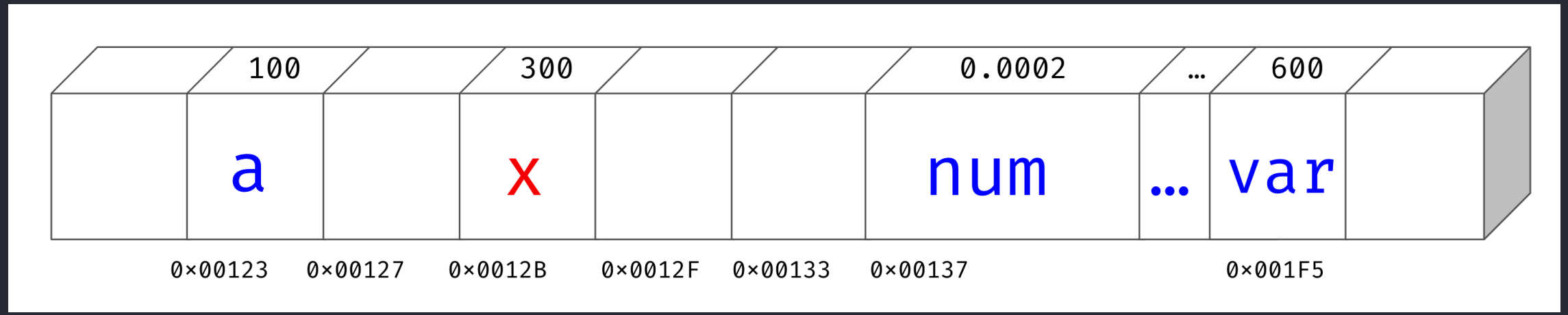
```
<type> name;
```

```
int x;
```

一般變數

```
<type> name;
```

```
int x;
```

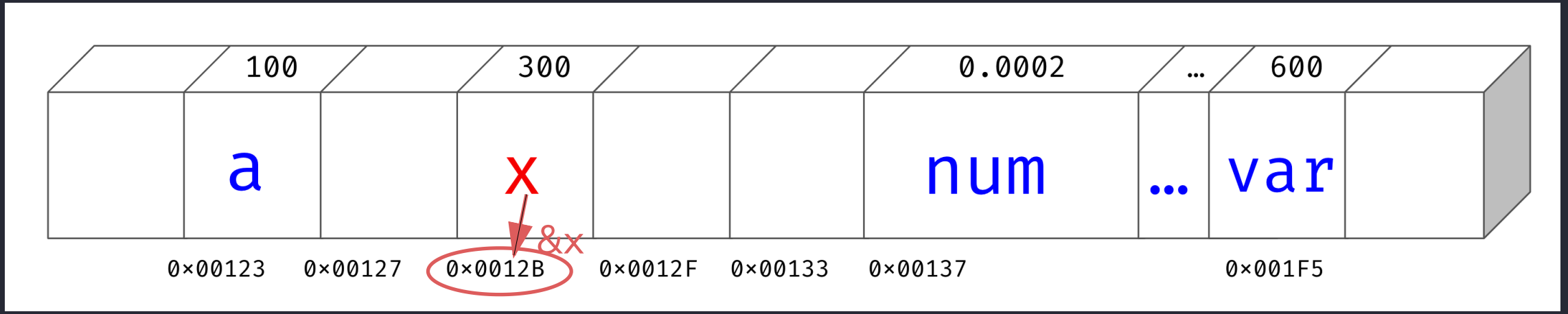


每個變數都有自己的記憶體位址。

一般變數

```
<type> name;
```

```
int x;
```



每個變數都有自己的記憶體位址。

印出指標變數位址

```c
#include <stdio.h>
int main(){
    int x;
    printf("%p", &x);

    return 0;
}
```

`%p` 是printf用來印出指標變數位址的格式指令字(format specifier)

指標變數

一種特殊的變數，它存儲的是一個記憶體位址

指標變數

```
<type>* name;
```

```
<type> *name;
```

```
int* p;
```

```
int *p;
```

指標變數

一種特殊的變數，它存儲的是一個記憶體位址

指標變數

```
<type>* name;
```

```
<type> *name;
```

```
int* p;
```

```
int *p;
```

一般變數

```
<type> name;
```

```
int x;
```

指標

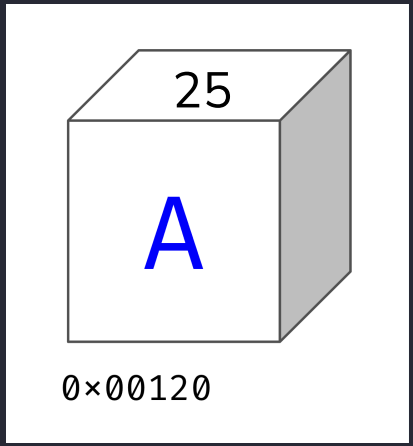
```c
#include <stdio.h>
int main(){
    int A = 25;
    int *ptrA; // 宣告一個int* 指標變數
    ptrA = &A; //A在記憶體中的位址
    return 0;
}
```

`&`：取址運算子

指標

```c
#include <stdio.h>
int main(){
    int A = 25;
    int *ptrA; // 宣告一個int* 指標變數
    ptrA = &A; //A在記憶體中的位址
    return 0;
}
```

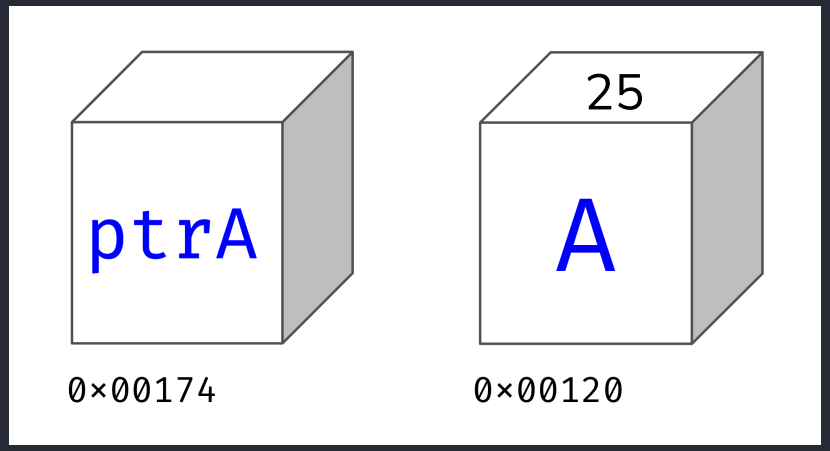
`&`：取址運算子



指標

```c
#include <stdio.h>
int main(){
    int A = 25;
    int *ptrA; // 宣告一個int* 指標變數
    ptrA = &A; //A在記憶體中的位址
    return 0;
}
```

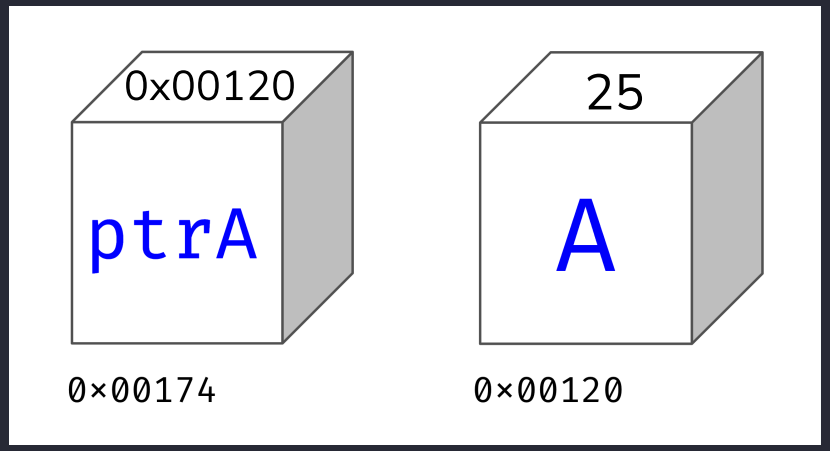
`&`：取址運算子



指標

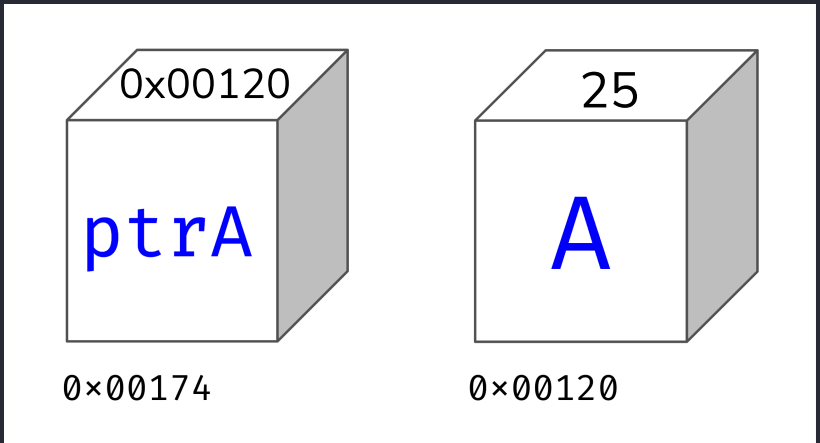
```c
#include <stdio.h>
int main(){
    int A = 25;
    int *ptrA; // 宣告一個int* 指標變數
    ptrA = &A; //A在記憶體中的位址
    return 0;
}
```

`&`：取址運算子



指標

```c
#include <stdio.h>
int main(){
    int A = 25;
    int *ptrA; // 宣告一個int* 指標變數
    ptrA = &A; //A在記憶體中的位址
    return 0;
}
```



`&`：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：0x00174

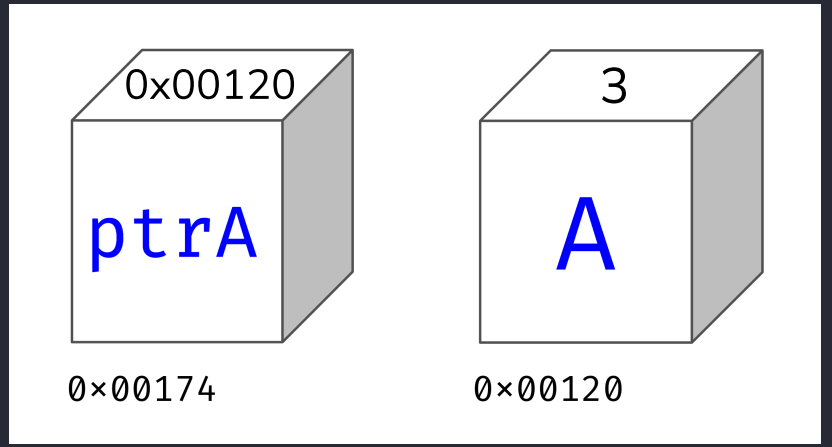
指標範例1

```c
#include <stdio.h>
int main(){
    int A = 3;
    int *ptrA = &A;

    printf("%d\n", *ptrA);
    *ptrA = 50;
    printf("%d\n", A);
}
```

`*` 取值運算子

`*ptrA` 就是 A



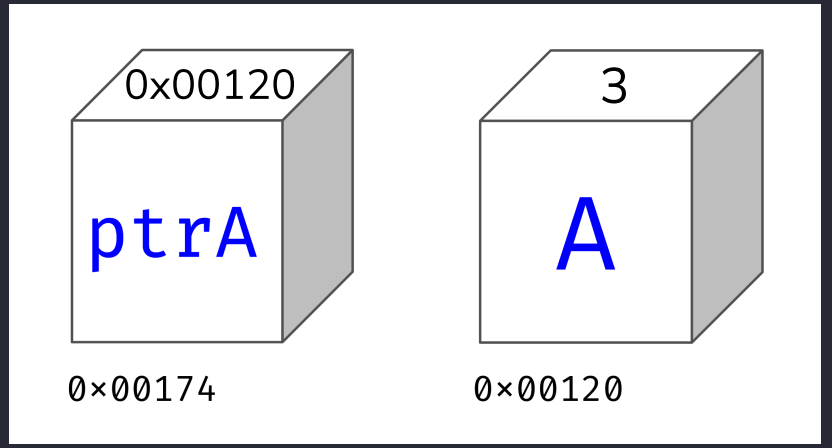
指標範例1

```c
#include <stdio.h>
int main(){
    int A = 3;
    int *ptrA = &A;

    printf("%d\n", *ptrA);
    *ptrA = 50;
    printf("%d\n", A);
}
```

`*` 取值運算子

`*ptrA` 就是 A



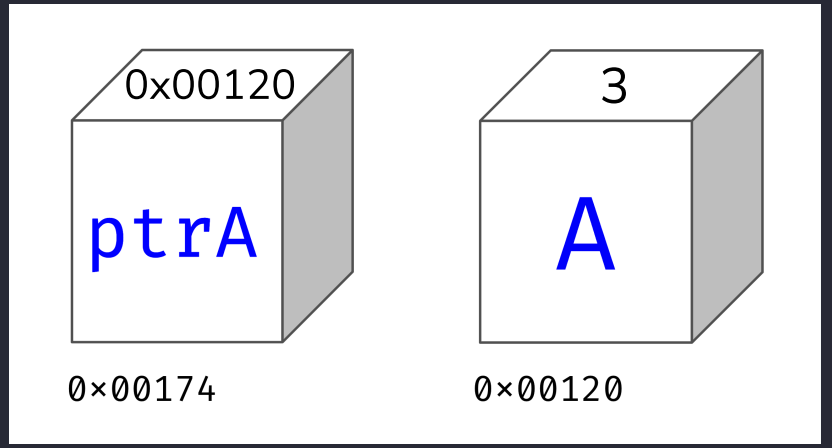
指標範例1

```c
#include <stdio.h>
int main(){
    int A = 3;
    int *ptrA = &A;

    printf("%d\n", *ptrA);
    *ptrA = 50;
    printf("%d\n", A);
}
```

`*` 取值運算子

`*ptrA` 就是 A



`*ptrA = 50`

透過(*ptrA)去更動A的值。

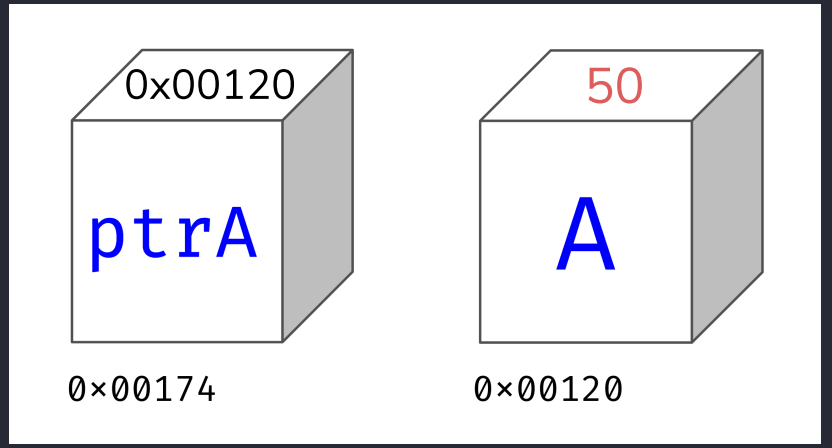
指標範例1

```c
#include <stdio.h>
int main(){
    int A = 3;
    int *ptrA = &A;

    printf("%d\n", *ptrA);
    *ptrA = 50;
    printf("%d\n", A);
}
```

`*` 取值運算子

`*ptrA` 就是 A



`*ptrA = 50`

透過(*ptrA)去更動A的值。

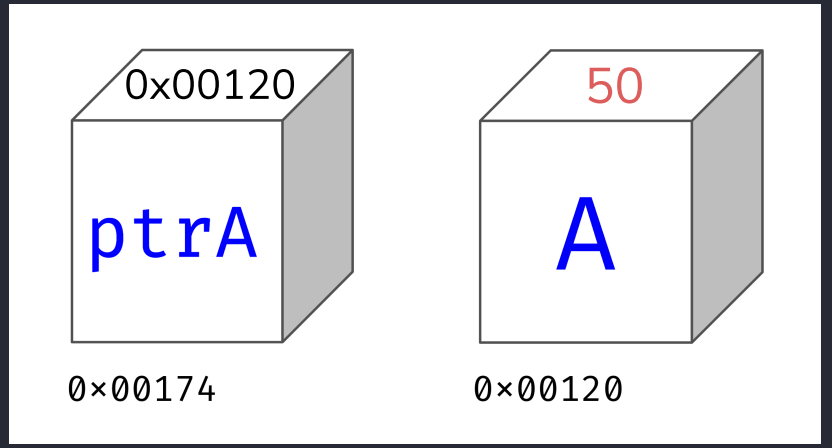
指標範例1

```c
#include <stdio.h>
int main(){
    int A = 3;
    int *ptrA = &A;

    printf("%d\n", *ptrA);
    *ptrA = 50;
    printf("%d\n", A);
}
```

`*` 取值運算子

`*ptrA` 就是 A



`*ptrA = 50`

透過(*ptrA)去更動A的值。

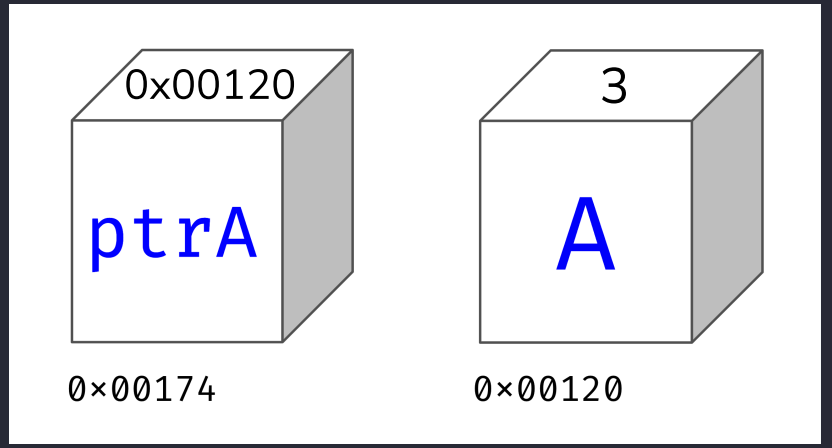
指標範例1

```c
#include <stdio.h>
int main(){
    int A = 3;
    int *ptrA = &A;

    printf("%d\n", *ptrA);
    *ptrA = 50;
    printf("%d\n", A);
}
```

`*` 取值運算子

`*ptrA` 就是 A



`*ptrA = 50`

透過(*ptrA)去更動A的值。

輸出：

```
3
50
```

`&` 符號

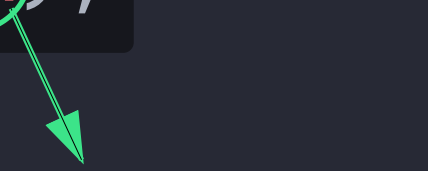
是不是似曾相似呢？

scanf 複習

```c
scanf("%d", &x);
```

scanf 複習

```
scanf("%d", &x);
```



取出x的位址

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+1` 是 `arr[1]` 的所在位址

`arr+2` 是 `arr[2]` 的所在位址

`*arr` 就是 `arr[0]`

`*(arr+1)` 就是 `arr[1]`

`*(arr+2)` 就是 `arr[2]`

指標與陣列 範例1

```c
#include <stdio.h>
int main(){
    int arr[5];
    for (int i=0; i<5; i++)
        scanf("%d", arr+i); // 和 &arr[i] 一樣
    for (int i=0; i<5; i++)
        printf("%d ", *(arr+i)); // 和 arr[i] 一樣
    printf("\n");
    return 0;
}
```

輸入：

```
3 7 2 9 4
```

指標與陣列 範例1

```c
#include <stdio.h>
int main(){
    int arr[5];
    for (int i=0; i<5; i++)
        scanf("%d", arr+i); // 和 &arr[i] 一樣
    for (int i=0; i<5; i++)
        printf("%d ", *(arr+i)); // 和 arr[i] 一樣
    printf("\n");
    return 0;
}
```

輸入：

```
3 7 2 9 4
```

指標與陣列 範例1

```c
#include <stdio.h>
int main(){
    int arr[5];
    for (int i=0; i<5; i++)
        scanf("%d", arr+i); // 和 &arr[i] 一樣
    for (int i=0; i<5; i++)
        printf("%d ", *(arr+i)); // 和 arr[i] 一樣
    printf("\n");
    return 0;
}
```

輸入：

```
3 7 2 9 4
```

指標與陣列 範例1

```c
#include <stdio.h>
int main(){
    int arr[5];
    for (int i=0; i<5; i++)
        scanf("%d", arr+i); // 和 &arr[i] 一樣
    for (int i=0; i<5; i++)
        printf("%d ", *(arr+i)); // 和 arr[i] 一樣
    printf("\n");
    return 0;
}
```

輸入：

```
3 7 2 9 4
```

指標與陣列 範例1

```c
#include <stdio.h>
int main(){
    int arr[5];
    for (int i=0; i<5; i++)
        scanf("%d", arr+i); // 和 &arr[i] 一樣
    for (int i=0; i<5; i++)
        printf("%d ", *(arr+i)); // 和 arr[i] 一樣
    printf("\n");
    return 0;
}
```

輸入：

```
3 7 2 9 4
```

輸出：

```
3 7 2 9 4
```

指標與陣列 範例2

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

    return 0;
}
```

指標與陣列 範例2

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

    return 0;
}
```

輸出：

```
3
10
```

用函式傳遞指標

```c
#include <stdio.h>
void swap(int* a, int* b){
    int t = *a;
    *a = *b;
    *b = t;
    return;
}
int main(){
    int x = 3, y = 5;
    swap(&x, &y);
    printf("%d %d\n", x, y);
    return 0;
}
```

輸出：

```
5 3
```

用函式傳遞指標練習

```c
#include <stdio.h>
int x = 3, y = 5;
void func(int *a, int *b){
    a = &x;
    b = &y;
    printf("%d %d\n", *a, *b);
}

int main(){
    int *a = &x, *b = &y;
    func(a, b);
    printf("%d %d\n", *a, *b);

    return 0;
}
```

輸出：

用函式傳遞指標練習

```c
#include <stdio.h>
int x = 3, y = 5;
void func(int *a, int *b){
    a = &x;
    b = &y;
    printf("%d %d\n", *a, *b);
}

int main(){
    int *a = &x, *b = &y;
    func(a, b);
    printf("%d %d\n", *a, *b);

    return 0;
}
```

輸出：

```
5 3
```

用函式傳遞指標練習

```c
#include <stdio.h>
int x = 3, y = 5;
void func(int *a, int *b){
    a = &x;
    b = &y;
    printf("%d %d\n", *a, *b);
}

int main(){
    int *a = &x, *b = &y;
    func(a, b);
    printf("%d %d\n", *a, *b);

    return 0;
}
```

輸出：

```
5 3
3 5
```

用函式傳遞陣列

```c
#include <stdio.h>
void printArr(int a[]){
    for(int i = 0; i < 5; i++){
        printf("%d ", a[i]);
    }
    printf("\n");
    return;
}

int main(){
    int arr[5] = {2, 1, 7, 4, 5};
    printArr(arr);
    return 0;
}
```

```c
#include <stdio.h>
void printArr(int* a){
    for(int i = 0; i < 5; i++){
        printf("%d ", a[i]);
    }
    printf("\n");
    return;
}

int main(){
    int arr[5] = {2, 1, 7, 4, 5};
    printArr(arr);
    return 0;
}
```

用函式傳遞字串

```c
#include <stdio.h>
void printStr(char* s){
    printf("%s\n", s);
    return;
}

int main(){
    char s[] = "Hello, World";
    printStr(s);
    return 0;
}
```

用函式傳遞二維陣列

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

Thanks for Listening