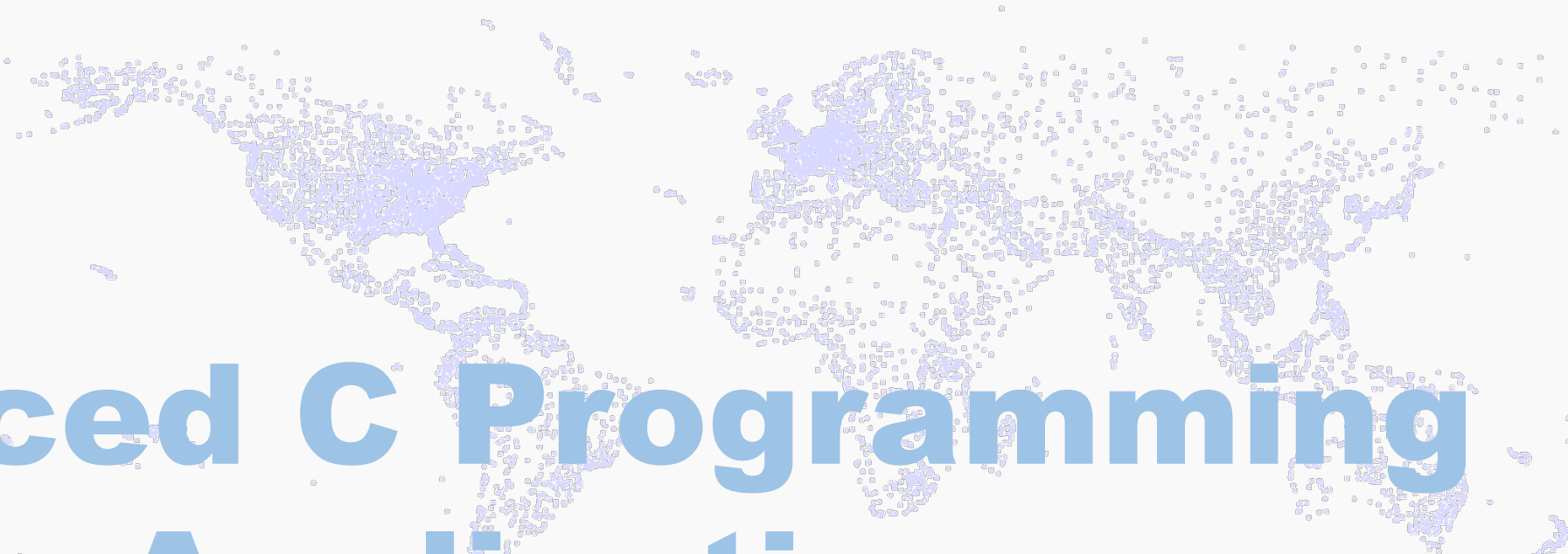


<Adv C & App/>



Advanced C Programming And It's Application

Pointer I: Pointer Declaration & Dereference

Assistant Prof. Chan, Chun-Hsiang

Department of Artificial Intelligence, Tamkang University

Oct. 27, 2021

</ Adv C & App >

大綱

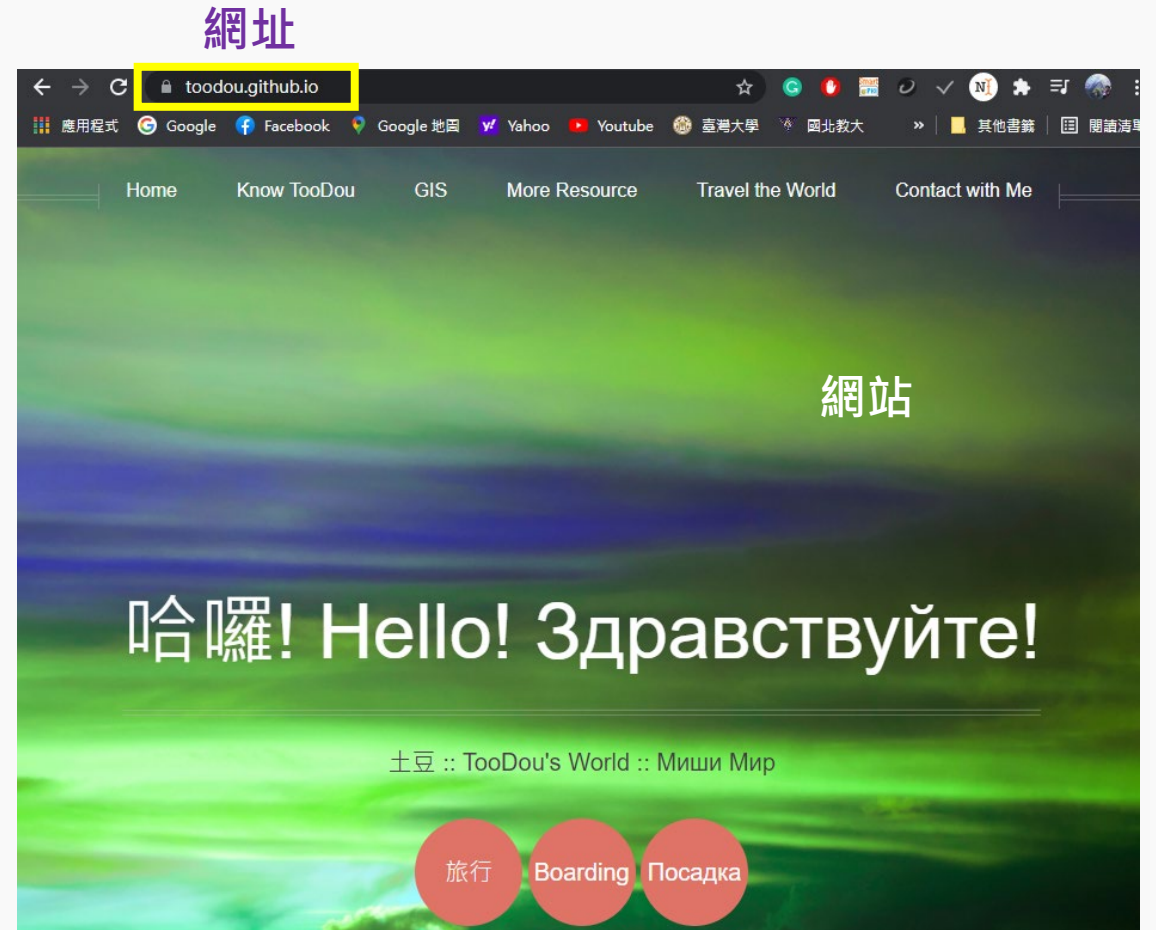
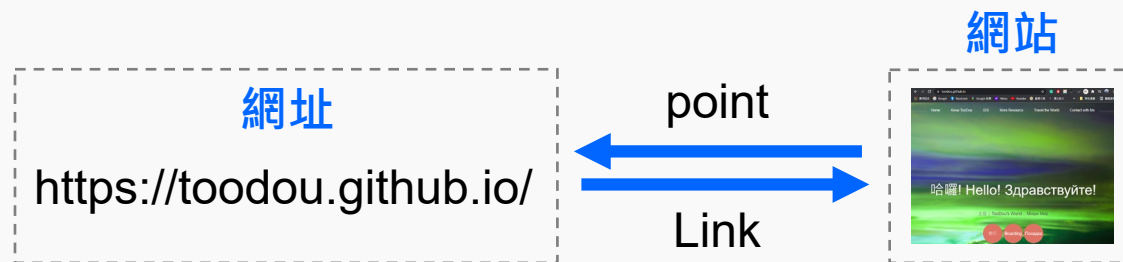
- [1] Concept
- [2] Declare a Pointer
- [3] Address
- [4] Dereference
- [5] Operators +/- in Pointers
- [6] Assignments



甚麼是指標？

用一個生活例子再來看一下，當我們要看一個網站的時候，可以用兩種方式去尋找。其一，直接用網址搜尋；其二，利用關鍵字來搜尋。

這件事告訴我們一件事，每一個網站都一定會有一個網址。



<Concept/>

變數地址

「變數地址」要如何取得，其實我們之前已經用過很多次了！在我們接使用者輸入的數值時，我們就是用「&」來呼叫地址，再把要輸入的值放到這個地址裡面。

```
/*Ex 5-1: Get Variable Address */  
printf("Ex 5-1: Get Variable Address\n");  
int a;  
  
printf("Enter an integer number!\n", a);  
scanf("%d", &a);  
printf("a = %d\n", a);  
printf("a's address is %p\n", &a);
```

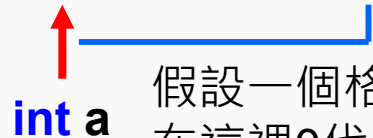
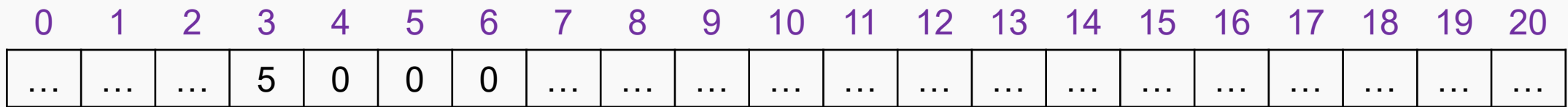
甚麼是指標?

身為一個底層程式語言，C/C++語言有一個特別的東西叫做「指標」，專門來存變數的記憶體地址。一般來說，我們宣告一個變數的時候，程式就會跟記憶體要一個儲存空間，來放你變數要給的數值。

我們可以將記憶體想像是一直列的格子。。

變數名稱
變數值
變數地址

```
int a = 5;
```



假設一個格子是1 byte，那麼int的空間是4 byte，所以要佔四格。
在這裡0代表NULL(空值)，數值會從最左邊開始填，超過255就會進位到下一格!

那如何宣告一個指標？

「指標」就跟一般的變數一樣，也有不同的資料類別。例如:儲存整數變數用的指標就是要用整數的指標。

```
/*Ex 5-2: Declare Pointer */  
printf("Ex 5-2: Declare Pointer\n");  
int a = 5;  
float b = 1.2;  
int *p;  
float *q;
```

將地址存到指標中

```
/*Ex 5-3: Get Variable Address2 */  
printf("Ex 5-3: Get Variable Address2\n");  
int a = 5;  
float b = 1.2;  
int *p = &a;  
float *q = &b;  
  
printf("a = %d (address: %p)\n", a, &a);  
printf("b = %f (address: %p)\n", b, &b);  
printf("p = %p\n", p);  
printf("q = %p\n", q);
```

Pointer's value & Pointer's address

當我們要宣告一個整數變數a時 → `int a`

當我們要知道整數變數a的數值的時候 → `a`

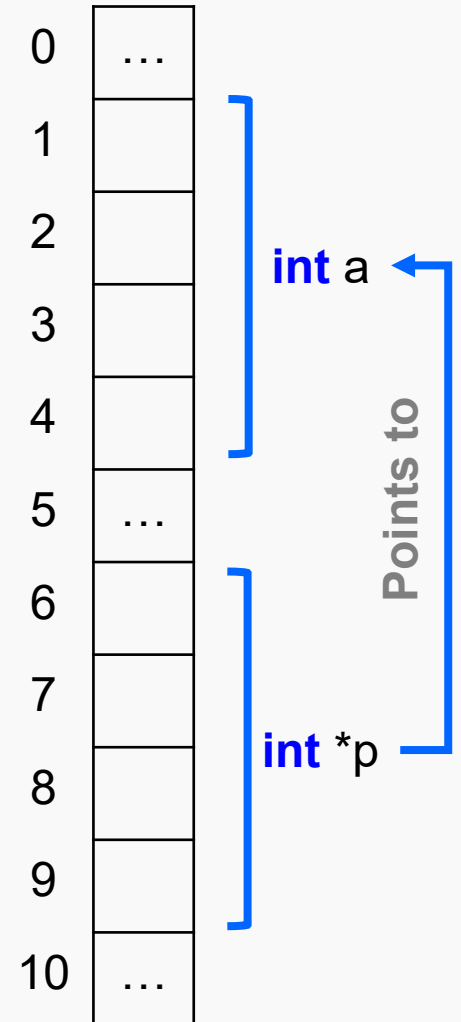
當我們要知道整數變數a地址時 → `&a`

→ 當我們要宣告一個整數指標p時 → `int *p`

→ 當我們要宣告一個整數指標p地址存的數值時 → `*p`

當我們要宣告一個整數指標p地址時 → `&p`

Dereference operator



<Dereference/>

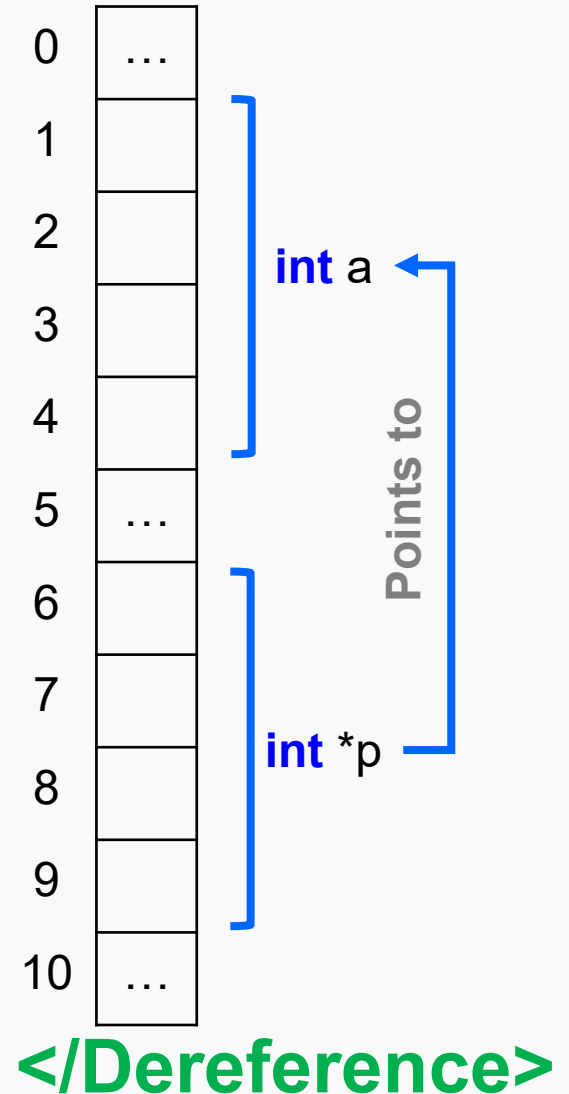
Pointer's value & Pointer's address

當我們要宣告一個整數指標p時 → `int *p`

→ 「*」是拿來宣告指標用的符號

當我們要宣告一個整數指標p地址存的數值時 → `*p`

→ 「*」是拿來取指標的數值用的operator



<Dereference/>

更改數值

當我們要改變數值的時候，其實除了直接針對我們的變數做修改以外，也可以用指標的方式做修改。

因為指標指向的是變數的地址，所以你可以直接去變數儲存的位置做更改。

如同旁邊的範例：

```
/*Ex 5-4: Get Variable Address2 */  
printf("Ex 5-4: Get Variable Address2\n");  
int a = 5;  
int *p = &a;
```

```
printf("a = %d (address: %p)\n", a, &a);  
printf("p = %p\n", p);
```

```
*p = 16;  
printf("*p = 16;\n");  
printf("a = %d (address: %p)\n", a, &a);  
printf("p = %p\n", p);
```

```
a = 512;  
printf("a = 512;\n");  
printf("a = %d (address: %p)\n", a, &a);  
printf("p = %p\n", p);
```

</Dereference>

Lab 5-1

試試看以下的程式碼，想一下為何會有error，寫出你覺得原因？

(a)

```
1  #include <stdio.h>
2
3  int main(){
4      int a = 5;
5      float b = 1.1;
6
7      int *p = a;
8      int *q = b;
9      return 0;
10 }
```

(b)

```
1  #include <stdio.h>
2
3  int main(){
4      int a = 5;
5      float b = 1.1;
6
7      int *p = &a;
8      int *q = &b;
9      return 0;
10 }
```

(c)

```
1  #include <stdio.h>
2
3  int main(){
4      int a = 5;
5      int b = 1;
6
7      int *p = &a;
8      int *q = &a;
9      *q = b;
10     p = a;
11     return 0;
12 }
```

Lab 5-2

寫出這些printf的結果，並寫出其原因!

(a)

```
1  #include <stdio.h>
2
3  int main(){
4      int a = 5;
5      int b = 1;
6      int *p = &a;
7      printf("a = %d; b = %d; *p = %d\n", a, b, *p);
8      int *q = &a;
9      *q = 12;
10     printf("a = %d; b = %d\n", a, b);
11     printf("*p = %d; *q = %d\n", *p, *q);
12 }
```

(b)

```
1  #include <stdio.h>
2
3  int main(){
4      int a = 5;
5      int b = 1;
6      int *p = &a;
7      printf("a = %d; b = %d; *p = %d\n", a, b, *p);
8      int *q = &a;
9      *q = 12;
10     printf("a = %d; b = %d\n", a, b);
11     printf("*p = %d; *q = %d\n", *p, *q);
12 }
```

Lab 5-2

寫出這些printf的結果，並寫出其原因!

(c)

```
1  #include <stdio.h>
2
3  int main(){
4      int a = 5;
5      int b = 1;
6      int *p = &a;
7      printf("a = %d; b = %d; *p = %d\n", a, b, *p);
8      int *q = &a;
9      *q = 12;
10     a = 16;
11     printf("a = %d; b = %d\n", a, b);
12     printf("*p = %d; *q = %d\n", *p, *q);
13     *p = 21;
14     printf("a = %d; b = %d\n", a, b);
15     printf("*p = %d; *q = %d\n", *p, *q);
16 }
```

Operators +/- in Pointers

還記得我們前面有提到，每個變數的指標在宣告的時候，要注意你要儲存變數的資料類別！那如果在地址上+1，那會怎麼變化呢？

```
/*Ex 5-5: Operator +/- in Pointers */
```

```
printf("Ex 5-5: Operator +/- in Pointers\n");
```

```
int a = 5;
```

```
int *p = &a;
```

```
printf("The values of a and its address are %d and %p, respectively.\n", a, &a);
```

```
printf("The values of p, p dereference and its address are %p, %d and %p, respectively.\n", p, *p, &p);
```

```
printf("\nPLUS 1 ::\n");
```

```
a = a + 1; // 觀察變數a加1的效果
```

```
p = p + 1; // 觀察地址加1的效果
```

```
printf("The values of a and its address are %d and %p, respectively.\n", a, &a);
```

```
printf("The values of p, p dereference and its address are %p, %d and %p, respectively.\n", p, *p, &p);
```

Operators +/- in Pointers

這邊以**Ex 5-5**我們稍微簡單簡介一下，Operators +/- 對指標的影響與記憶體空間內的16進位表示法：

```
Ex 5-5: Operator +/- in Pointers
The values of a and its address are 5 and 00000000061FE1C, respectively.
The values of p, p dereference and its address are 00000000061FE1C, 5 and 00000000061FE10, respectively.

PLUS 1 ::
The values of a and its address are 6 and 00000000061FE1C, respectively.
The values of p, p dereference and its address are 00000000061FE20, 13112560 and 00000000061FE10, respectively.
```

先前有提到int的大小是4 bytes，所以當我們直接在指標上加1的時候，就是跳到下一個可以儲存一個int大小的位置，也就是要跨過4個byte的記憶體位置。

Operators +/- in Pointers

那我們用圖解來看一下!

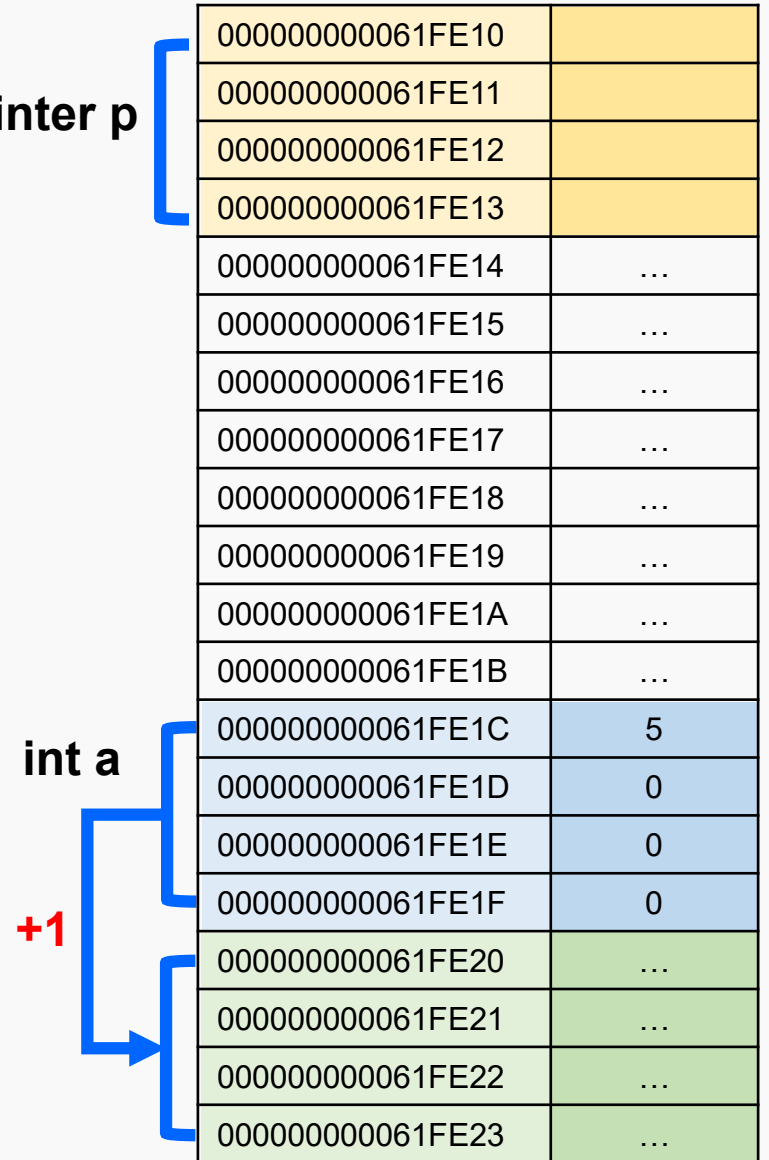
Before plus one,

Variable name	Value	Address
a	5	00000000061FE1C
p	00000000061FE1C	00000000061FE10

After plus one,

Variable name	Value	Address
a	6	00000000061FE1C
p	00000000061FE20	00000000061FE10

Address +1



Operators +/- in Pointers

還記得我們前面有提到，每個變數的指標在宣告的時候，要注意你要儲存變數的資料類別！那可曾想過如果我在地址上+1，那地址會怎麼變化呢？

Lab 5-3:

先分別宣告四個不同資料型別的變數，再用適合指標分別儲存他們的地址。再pointer + 1前後，記得印出以下四個需要的數值：

- (1) 變數值
- (2) 變數地址
- (3) 指標值
- (4) 指標地址

到底在記憶體空間中指標是什麼樣的存在

這邊是一題進階練習，大家可以回去動腦做做看：

Q: 想像看如果想要看到某個特定記憶體空間內的數值，要怎麼做呢？

目的: 讓我們看見在記憶體空間內的整數變數a與整數指標p的位置與所儲存的數值。

```
int a = 5;
```

```
int *p = &a;
```

範例結果:

```
==== Memory overview ====
000000000061FDF0    8  0  0  0  0  0  0  0  4 254
000000000061FDFA   97 0  0  0  0  0  8  0  0  0
000000000061FE04    5 0  0  0 10  0  0  0 20  0
000000000061FE0E    0 0  4 254 97  0  0  0  0  0
```

提示

這邊是一題進階練習，大家可以回去動腦做做看：

1. 先宣告一個整數變數與整數指標。
2. 將整數變數的地址與數值印出來。
3. 將整數指標儲存的地址與數值印出來。
4. 宣告一個字元指標儲存整數指標所存地址。
5. 定義一個掃描範圍，將該字元指標的前後地址與所存的數值回傳並印出。

解答

```
1 #include <stdio.h>
2
3 int main (){
4     /*Ex 5-6: Find your value*/
5     /* find pointer in your memory*/
6
7     // declare variable and pointer
8     int a = 5;
9     int *p = &a;
10
11    // print value and address
12    printf("Ex 5-6: Declare a Pointer - find pointer in your memory\n");
13    printf("(1) The value of a = %d\n", a);
14    printf("(2) The address of a in memory is %p\n", &a);
15    printf("(3) The value of pointer p is %p\n", p);
16    printf("(4) The pointing value of pointer p is %d\n", *p);
17    printf("(5) The address of pointer p is %p\n", &p);
18
19    // force integer pointer value store into a char pointer
20    unsigned char* q = (char*) p;
21    printf("(6) The value of pointer q is %p\n", q);
22    printf("(7) The pointing value of pointer q is %d\n", *q);
23    printf("(8) The address of pointer q is %p\n\n", &q);
24
25    // set search range
26    int range4search = 20;
27
28    // set number for newline
29    int newlineNum = 10;
30
31    // print all values of selected memory locations
32    printf("==== Memory overview ====");
33    for (int i=-range4search; i<range4search; i++){
34        if (i%newlineNum==0){
35            // print the first address and value
36            printf("\n%p\t %d\t", q+i, *(q+i));
37        }else{
38            // print value
39            printf("%d\t", *(q+i));
40        }
41    }
42 }
```


Lab 5-4

試試看將a的數值改成以下六種數字：將結果截圖分別儲存，並說明你在記憶體空間中，發現儲存變數a位置中，數值變化規律。

(1) 25

(2) 255

(3) 256

(4) 512

(5) 1024

(6) 1028

<Assignments/>

作業一

(1) 做兩個函數可以做 **SWAP** 的動作，宣告兩個整數 `int a` 與 `int b`，如果 $a < b$ ，則 `a` 與 `b` 數字交換。為了驗證數字確有交換，交換前後必須要印出 `a` 與 `b` 的數字。

You may do this without pointers!

(2) 接續第一小題，讓使用者輸入兩個整數，並將交換這個過程寫進一個獨立的 `int swap_val()` 函數。

You cannot do this without pointers!

```
(1) void main(){
    // declare int a and int b
    // print a and b
    // swap
    // print a and b
}

(2) void swap(){
    //swap
}
int main(){
    // declare int a and int b
    // scan a and b
    // print a and b
    // call swap()
    // print a and b
}
```

<Reference/>

參考文獻

- <https://kopu.chat/2017/05/15/c%E8%AA%9E%E8%A8%80-%E8%B6%85%E5%A5%BD%E6%87%82%E7%9A%84%E6%8C%87%E6%A8%99%EF%BC%8C%E5%88%9D%E5%AD%B8%E8%80%85%E8%AB%8B%E9%80%B2%EF%BD%9E/>