



Pointer & Memory Management 指標與記憶體管理

主講人：資訊二甲 林柏丞
日期：2014 / 12 / 17

C 語言指標 (Pointer)

指標是儲存記憶體位址的資料型態
如果想知道變數的記憶體位址為何，
可以使用&運算子，&是「取址運算子」
(Address-of operator)，它可以取出變數的記憶體位址。

C 語言指標 (Pointer)

指標擁有兩種操作特性，一是操作指標所儲存的位址，一是操作指標所指向位址之資料，可以使用取值 (Dereference) 運算子*來提取指標所指向位址的資料。

指標範例

```
int var = 12;  
int *ptr1 = &var; // int *a, *b; ( 都是指標 )  
int* ptr2 = &var; // int* a,b; ( b不是指標 )  
int *ptr3;  
ptr3 = &var ;
```

```
printf("var = %d\n" , var);  
printf("ptr1 = %p\n" , ptr1);  
printf("ptr2 = %p\n" , ptr2);  
printf("ptr3 = %p\n\n" , ptr3);
```

```
printf("*ptr1 = %d\n" , *ptr1);  
printf("&ptr1 = %p\n" , &ptr1);
```

名稱	位址	內容
var	100	12
Ptr1	104	100
ptr2	108	100
Ptr3	112	100
	116	?

初始化指標

宣告指標但不指定初值，則指標指向的位址是未知的，存取未知位址的記憶體內容是危險的。

```
int *ptr;  
*ptr = 12;
```

初始化指標

初始化指標最好是指向NULL或不指向任何位址。

```
int *ptr = NULL;
```

```
int *ptr2 = 0;
```


指標運算

指標除了可以用一般運算外，還可以用運算記憶體位址。

```
int *ptr = 0;
```

```
ptr ++;
```

```
printf("ptr + 1 : %p\n", ptr + 1);
```

指標陣列

陣列是一連續記憶體空間，陣列名稱即指向陣列開頭，陣列索引即指標位移量。

```
int arr[10] = {0};  
int *ptr = arr;
```


指標陣列

把記憶體位址print出來：

&arr[0]: 100 ptr + 0: 100

&arr[1]: 104 ptr + 1: 104

- 若是取值則必須這樣寫：

arr[0]: 10 *(ptr + 0): 10

arr[1]: 20 *(ptr + 1): 20

宣告陣列

陣列使用的一個缺點，就是大小必須事先決定好，然而有時候無法知道我們會使用多大的陣列，或者希望由使用者自行決定陣列大小，這時候就可以使用動態記憶體配置加上指標運算來解決這個問題，先說明陣列動態配置的方式：

```
int *arr = malloc(arrSize * sizeof(int));
```

memset

```
char str[] = "always memset" ;  
memset(str, ' 0' ,sizeof(str)/sizeof  
(char));  
puts(str);
```

會印出：000...00。

雙重指標

指標指向一個指標，為雙重指標。

指標應用

使用函式時，若不使用全域變數，該如何從main或A函式修改B函式中變數的值呢？

答：使用指標傳遞變數位址。

練習

請設計一函式，可以交換變數a以及變數b的值。

Ex：

$a = 4, b = 10$

使用函式`swap(&a,&b);`後

$a = 10, b = 4$ 。

動態記憶體配置

C語言動態記憶體配置基本程序：

- 1.使用 malloc 等函式配置記憶體
- 2.依程式需要使用所配置的記憶體
- 3.使用 free 函式釋放記憶體

動態記憶體配置

```
int *pi = (int*) malloc(sizeof(int));  
*pi = 5;  
printf( "%d\n" ,*pi);  
free(pi);
```

以上程式會顯示5。建議使用sizeof增加可攜性。

malloc

原型 `void* malloc(size_t);`

配置一個int大小的空間：

```
int *pi = (int*) malloc(sizeof(int));
```

若記憶體不足，則傳回NULL。

malloc只負責配置，不會初始化。故目前ptr指向的位址內容是未知，必須手動初始化。

malloc

malloc會傳回void指標。void指標能指派給其他類型指標。所以不需要轉型，但轉型是比較好的做法。

- 1.能記錄 malloc 函式的目的
- 2.讓程式碼相容於C++以及早期C compiler。

free

使用malloc所配置的記憶體空間並不會自動還給OS。當不需要使用這塊空間時，必須以free函式將其釋放：

```
free(ptr);
```

free並不會檢查指標是否指向NULL。所以最好自己寫一個函式判斷是否為NULL。

calloc

`int *arr = malloc(1000 * sizeof(int));`這段程式碼配置了1000個int大小的空間，並傳回空間的第一個位址，配置後的空間內容是未知的。

若使用`calloc()`來配置空間則可以初始化內容為0：

```
int *arr = calloc(1000, sizeof(int));
```


記憶體洩漏

記憶體已不再使用卻沒有釋放：

- 1.遺失記憶體位址
- 2.應該呼叫卻沒有呼叫free函式

- 這些記憶體無法回收供後續使用，最後可能因為 malloc 無法配置記憶體而導致程式中止。

練習

請寫一程式，利用指標判斷是否迴文。

abcba → 迴文

• abcbba → 不迴文