

# 計算機程式設計

## C語言 Pointer

郭忠義

[jykuo@ntut.edu.tw](mailto:jykuo@ntut.edu.tw)

臺北科技大學資訊工程系

# 指標

- ❑ **指標變數** ptr 指向(儲存)變數 num 記憶體位址。
  - 指標變數的資料型別規定要在一般資料型別加上\*
  - 才可以存變數記憶體位址
  - 變數的記憶體位址由編譯器、作業系統配置。
  - &運算子，取出變數被編譯器、作業系統所配置的記憶體位址
  - \*運算子，取出指標變數所指向記憶體位址，裡面存的值
    - ptr指向 num 記憶體位址，num裡面存100。

二行可以合併寫成：

```
int *ptr=&num;
```

```
int main() {  
    int num;  
    num = 100;      // int num =100;  
    int *ptr;       // int *ptr = &num;  
    ptr = &num;     // *ptr == num  
    printf("num=%d, ptr=%p", num, ptr); //100, 1015  
    return 0;  
}
```

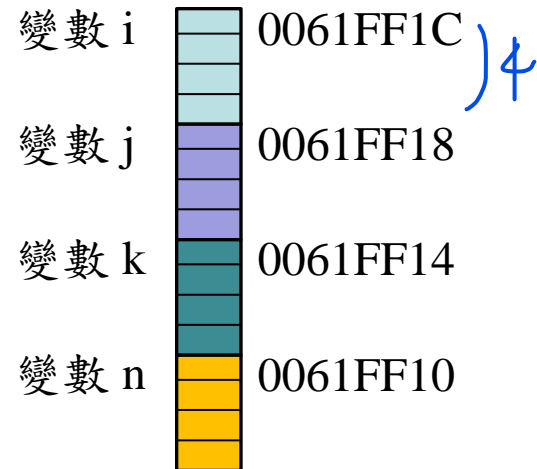
值	變數名	記憶體位址 (假設)
		1001
		...
100	num	1015
1015	ptr	1019
		...

# 記憶體位址

□ 印出變數的記憶體位址。

```
int main() {  
    int i=1, j=2, k=3, n=4;  
    printf("i 記憶體位置=%p\n", &i);  
    printf("j 記憶體位置=%p\n", &j);  
    printf("k 記憶體位置=%p\n", &k);  
    printf("n 記憶體位置=%p\n", &n);  
    return 0;  
}
```

i 記憶體位置=0061FF1C  
j 記憶體位置=0061FF18  
k 記憶體位置=0061FF14  
n 記憶體位置=0061FF10



# 記憶體大小

## □ 印出變數的記憶體位址。

```
01 01 char c = '1'; int i = 1;  
02 02 float f = 1.0f; double d = 1.0f;  
03 03 printf("c 記憶體位大小%d\n", sizeof(c));  
04 04 printf("i 記憶體位大小%d\n", sizeof(i));  
05 05 printf("f 記憶體位大小%d\n", sizeof(f));  
06 06 printf("c 記憶體位大小%d\n", sizeof(d));
```

c 記憶體位大小1  
i 記憶體位大小4  
f 記憶體位大小4  
c 記憶體位大小8

```
01 char c = '1'; int i = 1;  
02 float f = 1.0f; double d = 1.0f;  
03 char *cPtr = &c; int *nPtr = &i;  
04 float *fPtr = &f; double *dPtr = &d;  
05 printf("cPtr 記憶體位大小%d\n", sizeof(cPtr));  
06 printf("nPtr 記憶體位大小%d\n", sizeof(nPtr));  
07 printf("fPtr 記憶體位大小%d\n", sizeof(fPtr));  
08 printf("dPtr 記憶體位大小%d\n", sizeof(dPtr));
```

64bit 作業系統  
(編譯器)

cPtr 記憶體位大小8  
nPtr 記憶體位大小8  
fPtr 記憶體位大小8  
dPtr 記憶體位大小8

# 設定指標的值

```
01 int *ptr;  
02 *ptr = 35;
```



錯誤的指標初始值(記憶體位址)設定方法。  
記憶體位址由編譯器、作業系統配置。

```
01 int number = 100;  
01 int *ptr;  
03 ptr = &number;  
04 *ptr = 35;  
05 printf("*ptr = %d", *ptr);  
06 printf("number = %d", number);
```

輸出結果:  
\*ptr = 35  
number = 35

# Exercise



## □ 以下輸出

```
int i = 4, *p = &i, *q = &i;
printf("%d %d\n", *p, *&i);
i += (*q) * (*p);
printf("%d %d\n", i, q);
```

```
double i = 4, j = 6, *p = &j, *q = &i, *r;
printf("%f %f\n", *p, *q);
r = p; p = q; q = r;
printf("%f %f\n", *p, *q);
```

```
int i, j = 21, *p = &j, *q = p;
printf("%d %d\n", *p, *q);
for (i = 0; i < 4; i++)
    *(p++);           // ==> p++
(*q)++;
printf("%d %d %d", *p, *q, i);
```

```
int i = 4, j, number = 3, *p = &number;
for (j = 0; j < number; j++)
    (*p) += i--;
printf("%d %d %d %d\n", *p, number, j, i);
```

# 指標的轉型

```
void test() {  
    int x=1001, *p;  
    p=x;  
    printf("%d", *p);  
}
```

錯誤結果，p指向記憶體位址為1001的地方，  
但位址為1001，**裡面存的值未知**。

值	變數名	記憶體位址
?		1001
		...
1001	x	1015
1001	p	1019

```
void test() {  
    int x=1001, *p;  
    *p=x;  
    printf("%d", *p);  
}
```

可能造成系統問題，p指向記憶體位址未知，  
**在未知的位址內**，存入1001。

值	變數名	記憶體位址
		1001
		...
1001	x	1015
?	p	1019
		...
1001		?

# 指標的轉型

```
01 int *ptr;  
02 float f = 100.01f;  
03 ptr = &f;  
04 printf("=%0.2f", *(float *)ptr);
```

[Execution Error] \*ptr value is unexpected.

```
void *ptr;  
float f = 100.01f;  
ptr = &f;  
printf("=%0.2f", *(float *)ptr);
```

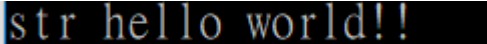
若想將指標指向不特定型別變數，  
可使用 **void** 型別宣告。可指向任何  
型別，編譯器不會發出警告。



# 字串常數

## ❑ 字串常數表示。

```
#define MAX "max"
int main() {
    const char* str = "str";
    printf("%s hello world!!", str);
    return 0;
}
```



相同Global記憶體空間  
輸出一樣

```
#include <stdio.h>
void f() {
    const char* str1 = "test";
    const char* str2 = "test";
    printf("%d\n%d\n", str1, str2);
}
int main() {
    const char* str = "test";
    f();
    printf("%d", str);
    return 0;
}
```

- 編譯階段在Global區段建立資料空間，生命週期和程式共存亡。
- 程式執行時，不論在任何地方取得字串常數都有效。
- 字串常數一旦建立，在程式執行中只能讀，不可被修改。

## ❑ 非字串常數

```
char str[] = "hello"; // (使用字串常數初始一個字串陣列)
char str[] = {'h', 'e', 'l', 'l', 'o', '\0'}; // 字串是字元陣列 + 字串結束符號
```

# 字串常數

## □ 字串常數。

- 不可透過指標解參考(dereference \*)**修改某個字元**。
- 指向字串常數的指標，若被設定指向另一個字串常數或字串陣列，或任何記憶體位址後，就無法再指向本來字串常數，因已**遺失字串常數的位址**。

```
#include <stdio.h>
int main() {
    char *p = "lose";
    char *str = "test";
    p = str;    // "lose"字串遺失
    str++;      // 指標可以移動
    (*str) = 'p'; // 不可以指標解參考修改，
                // 執行會錯誤，中斷執行
    printf("%c", *str);
    return 0;
}
```

```
#include <stdio.h>
int main() {
    char *p = "lose";
    printf("%c", *p); // 印出一個字元
    printf("%s", p);  // 印出一個字串
    return 0;
}
```

# 傳遞指標到函數

□ 可以讓函式回傳兩個值以上

```
01 void swap(int a, int b){  
02     int temp = b;  
03     b = a;  
04     a = temp;  
05 }  
06  
07 int main(){  
08     int a = 10, b = 12;  
09     swap(a, b);  
10     printf("a=%d, b=%d", a, b);  
11     return 0;  
12  
13 }
```

a=10, b=12

```
void swap(int *a, int *b){  
    int temp = *b;  
    *b = *a;  
    *a = temp;  
}  
  
int main(){  
    int a = 10, b = 12;  
    swap(&a, &b);  
    printf("a=%d, b=%d", a, b);  
    return 0;  
}
```

a=12, b=10

# Exercise

## □ 寫出以下輸出

```
void f(int a, int *b, int *c) {  
    int d; a = 2;  
    *b = 3; c = &a; d = 5;  
}  
int main() {  
    int a=1,b=2,c=3,d=4;  
    f(a, &b, &c);  
    printf("%d %d %d %d\n", a, b, c, d);  
    return 0;  
}
```

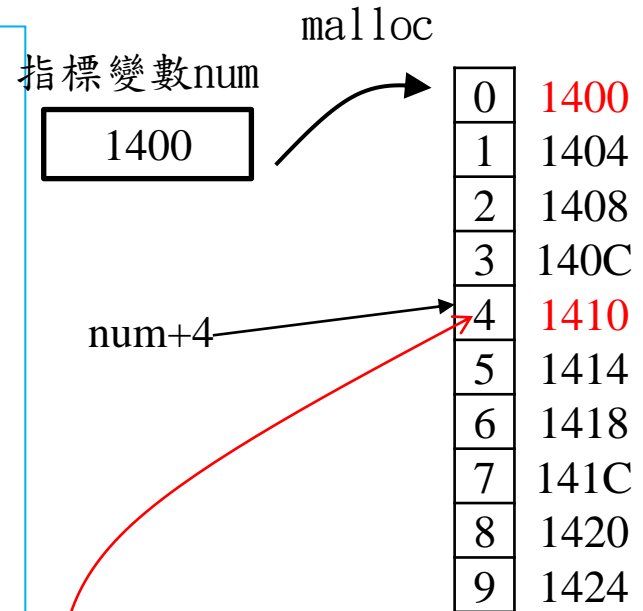
```
void function(int *a,int *b,int *c){  
    int *temp=a;  
    *b=(*c)*(*temp);  
    *c=*temp;  
    *a=10;  
    a=b;  
    b=c;  
    c=temp;  
}  
void main(void){  
    int i=-1,j=4,k=2,*p=&i,*q=&j;  
    function(p, q, &k);  
    printf("%d %d %d\n",i,j,k);  
}
```

# 動態記憶體配置

語法:(資料型別 \*)malloc(sizeof(資料型別) \* 個數)

```
#include <stdio.h>
int main(){
    int *num = (int *)malloc(sizeof(int)*10);
    if (num == NULL) exit(1);
    for(int i=0; i<10; i++) *(num+i) = i;
    for(int i=0; i<10; i++) printf("%d\n", *(num+i));
    free(num);
}
```

不再使用的空間必須還回系統



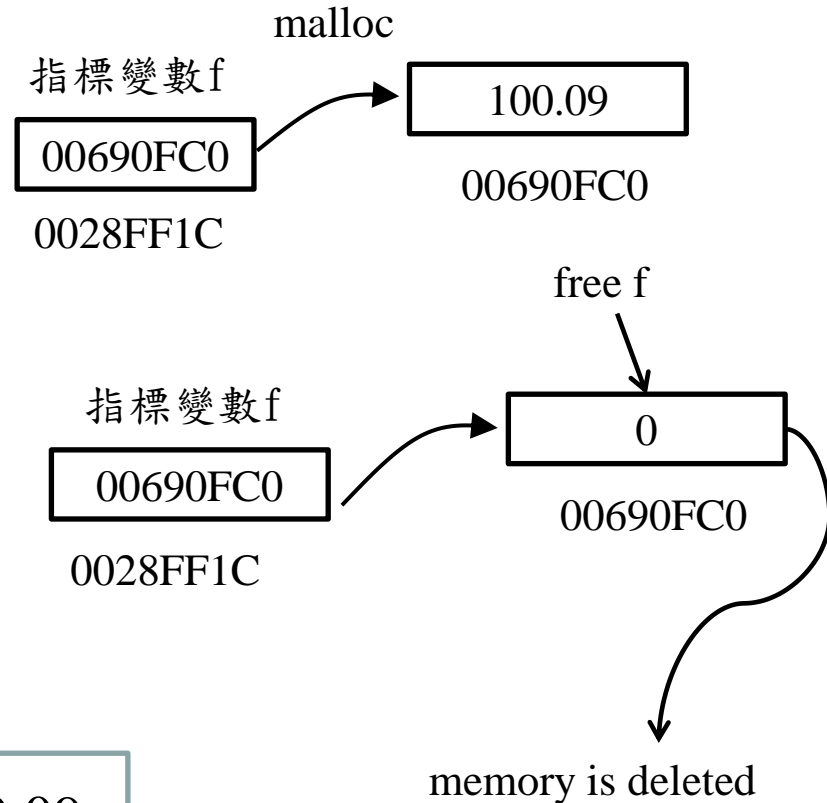
\* (num+4) , 是num+4指向記憶體內存的值

# 動態記憶體配置

❑ 印出變數的記憶體位址。

```
#include <stdio.h>
int main(){
    float *f = (float *)malloc(sizeof(float));
    *f = 100.09;
    printf("f=%p &f=%p *f=%.2f\n", f, &f, *f);
    free(f);
    printf("f=%p &f=%p *f=%.2f\n", f, &f, *f);
}
```

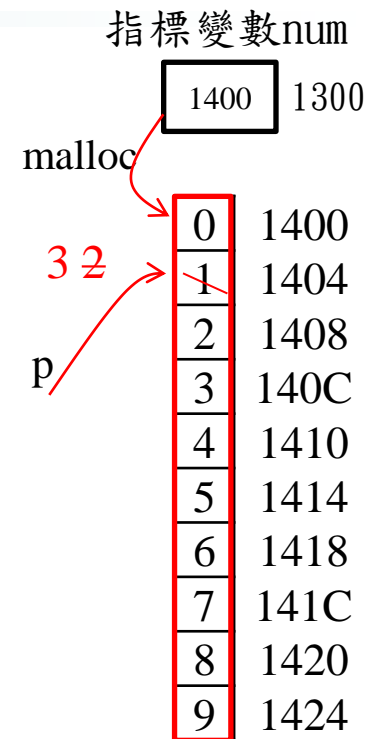
```
f=00690FC0 &f=0028FF1C *f=100.09
f=00690FC0 &f=0028FF1C *f=0.00
```



# 動態記憶體配置

## □ 指標++。

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int *p;
    int *num = (int *)malloc(sizeof(int)*10);
    for(int i=0; i<10; i++) *(num+i) = i;
    p = num;
    p++;
    printf("%d, ", (*p)++); // p 值不會變，取1後變2
    printf("%d, ", ++(*p)); // p 值不會變，2變3，取3
    printf("%d, ", *(p++)); // p 值會變，取3，p指向下一個 2
    printf("%d, ", *p++); //取2，p再 +1
    printf("\n%d, %d, %d, %d", num[0], num[1], num[2], num[3]);
    free(num);
}
```



1	3	3	2
0	3	2	3

# Exercise

## □ 分數輸入

○ 輸入 3/5，輸出 3, 5

```
void scan_fraction(int *nump, int *denomp) {
    char slash; /* character between numerator and denominator */
    int status; /* status code returned by scanf indicating number of valid values obtained */
    int error; /* flag indicating presence of an error */
    char discard; /* unprocessed character from input line */
    do { /* No errors detected yet */
        error = 0;
        /* Get a fraction from the user */
        printf("Enter a common fraction as two integers separated ");
        printf("by a slash> ");
        status = scanf("%d %c%d", _____, _____, _____);
        /* Validate the fraction */
        if (status < 3) {
            error = 1;
            printf("Invalid-please read directions carefully\n");
        }
    }
```



# Exercise

## □ 分數輸入

○ 輸入 3/5，輸出 3, 5

```
else if (slash != '/') {
    error = 1;
    printf("Invalid-separate numerator and denominator");
    printf(" by a slash (/)\n");
} else if (*denomp <= 0) {
    error = 1;
    printf("Invalid—denominator must be positive\n");
}
/* Discard extra input characters */
do {
    scanf("%c", &discard);
} while (discard != '\n');
} while (error);
}
```

# Homework I

## □ 分數運算

### ○ 輸入1

- `int n1, d1 /* 第一個分數的分子與分母*/`
- `int n2, d2 /* 第二個分數的分子與分母*/`
- `char op /* 數學運算 + - * or / */`
- `char again /* y or n 是否繼續 */`

### ○ 輸出1

- `int n_ans /* 答案分子 */`
- `int d_ans /* 答案分母 */`

### ○ 輸入2

- `I(n/d)opI(n/d)`，例如 `-2(2/3)*3(1/4)`

### ○ 輸出2

- `I(n/d)`，`-6(1/6)`

# Homework II

## □ 輸入平面上兩個點，求直線方程式

- 輸入兩個點，整數， $(x1, y1), (x2, y2)$
- 輸出  $y = mx + b$ 
  - $m = (y1 - y2) / (x1 - x2)$
  - $b = (x2y1 - x1y2) / (x2 - x1)$

```
int main() {  
    equation(1,0,0,-1);  
    equation(1,0,0,1);  
    equation(1,1,2,2);  
    equation(1,1,2,4);  
    equation(2,3,4,5);  
    equation(0,1,3,3);  
    return 0;  
}
```

```
y = x - 1  
y = - x + 1  
y = x  
y = 3 x - 2  
y = x + 1  
y = 2/3 x + 1
```

# 指標的指標

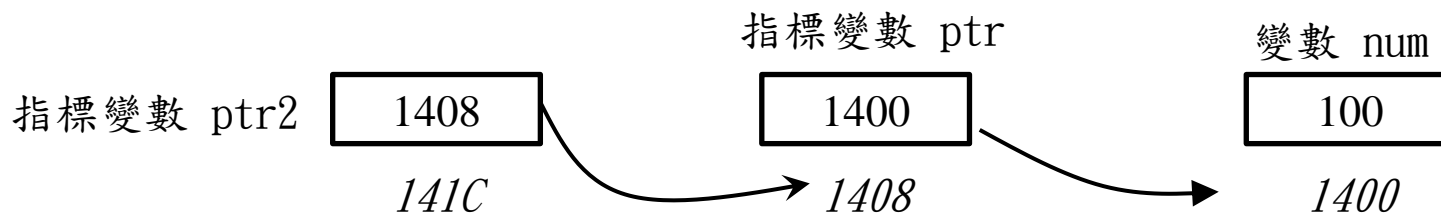
```
01 int num = 100;
02 int *ptr;
03 int **ptr2
04 ptr = &num;
05 ptr2 = &ptr;
06 printf("num=%d &num=%p\n", num, &num);
07 printf("*ptr=%d ptr=%p\n", *ptr, ptr);
08 printf("**ptr2=%d *ptr2=%p ptr2=%p\n", **ptr2, *ptr2, ptr2);
```

輸出結果:

num=100 &num=0028FF18

\*ptr=100 ptr=0028FF18

\*\*ptr2=100 \*ptr2=0028FF18 ptr2=0028FF14



# Exercise

## □ 寫出以下輸出

```
void f(int **a,int **b, int **c,int *d){
    *a=d;
    **b=(**a)/(**c);
    *d=(**b)*(**a);
}
void g(int **a, int **b, int **c, int *d){
    int **temp=a;
    a = b;    b = temp;
    *a = d;
    (**a) = (**b)+(**c);
}
void testF(){
    int i=5,j=-2,k=9,*p=&i,*q=&j,*x=&k;
    f(&p,&q,&x,&k);
    printf("%d %d %d",i,j,k);
}
```

```
void testG(){
    int i=5,j=-2,k=9,*p=&i,*q=&j,*x=&k;
    g(&p,&q,&x,&k);
    printf("%d %d %d",i,j,k);
}
```