第十章
指 標

記憶體位址與指標

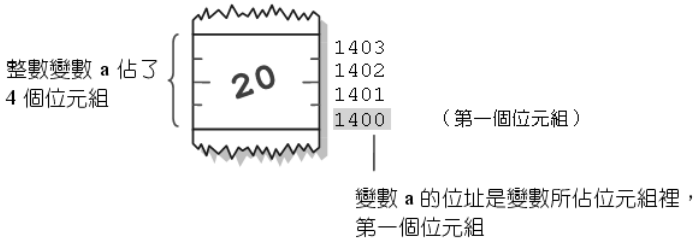
間接參考運算子的用法

指標與函數參數

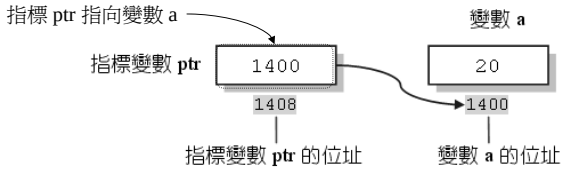
指標與陣列之間的關係

變數的位址

- 變數的位址是它所佔位元組裡，第一個位元組的位址：



- 指標變數是用來存放變數在記憶體中的位址



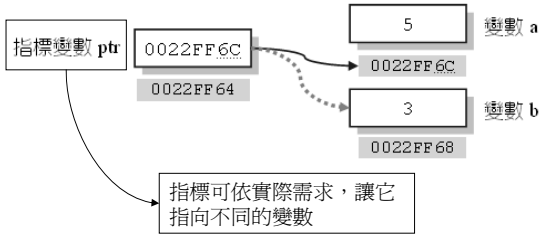
爲什麼要用指標？

- 利用指標可以使得函數在傳遞陣列時更有效率

```
int showArray(int arr[]) {
  ...
}
```

傳遞陣列的位址，而非整個陣列，因而執行速度較快

- 可隨時改變所指之對向，因而可彈性地
  - 處理不同的變數
  - 呼叫不同的函數

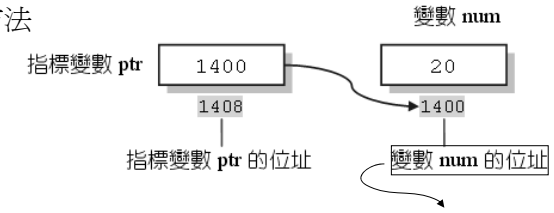


指標變數的宣告

指標變數的宣告

```
資料型態 *指標變數;
或
資料型態* 指標變數;
```

- 指標變數的使用方法
  - int num=20;
  - int *ptr;
  - ptr=#



第二行與第三行敘述也可以合併寫成：

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

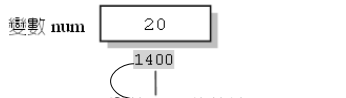
指標變數的使用

- 指標變數常見的用法有兩種：
  - 存取 指標變數裡 某一變數的位址
  - 存取 指標變數所指向之變數 的內容

address-of operator

位址運算子「&」

```
int num=20;
```



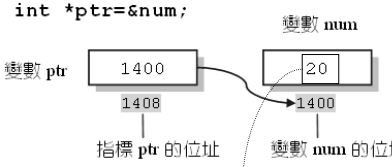
&num可得到變數 num 的位址，即 1400
*(&num) 即爲變數 num，不在等號左手邊時即爲num裡存放的數值 20

dereference operator or Indirection operator

依址取值運算子「*」

```
int num=20;
int *ptr=&num;
```

*ptr = 15;



*ptr 即爲 ptr 所指向之變數 num ，使用時不在等號左手邊時即爲num裡存放的數值 20

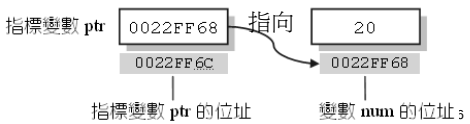
指標變數的使用範例 (1/2)

```
01  /* prog10_2, 指標變數的宣告 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int *ptr,num=20;
07
08     ptr=&num;
09     printf("num=%d, &num=%p\n",num,&num);
10     printf("*ptr=%d, ptr=%p, &ptr=%p\n",*ptr,ptr,&ptr);
11     system("pause");
12     return 0;
13  }
```

```
/* prog10_2 OUTPUT--------------------
num=20, &num=0022FF68
*ptr=20, ptr=0022FF68, &ptr=0022FF6C
---------------------------------------*/
```

執行完第6行後

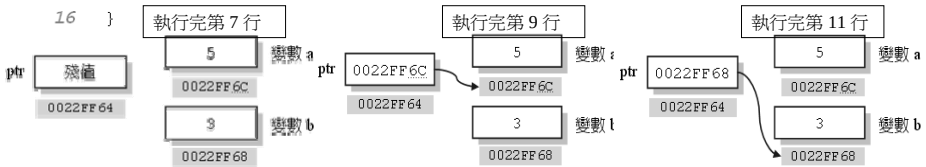
執行完第8行後



指標變數的使用範例 (2/2)

```
01  /* prog10_3, 指標變數的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int a=5,b=3;
07     int *ptr;
08
09     ptr=&a;              /* 將 a 的位址設給指標 ptr 存放 */
10     printf("&a=%p, &ptr=%p, ptr=%p, *ptr=%d\n",&a,&ptr,ptr,*ptr);
11     ptr=&b;              /* 將 b 的位址設給指標 ptr 存放 */
12     printf("&b=%p, &ptr=%p, ptr=%p, *ptr=%d\n",&b,&ptr,ptr,*ptr);
13
14     system("pause");
15     return 0;
16  }
```

```
/* prog10_3 OUTPUT-----------------------------
&a=0022FF6C, &ptr=0022FF64, ptr=0022FF6C, *ptr=5
&b=0022FF68, &ptr=0022FF64, ptr=0022FF68, *ptr=3
-----------------------------------------------*/
```



指標變數所佔的位元組

- 查詢指標變數所佔的位元組：

```
01  /* prog10_4, 指標變數的大小 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int *ptri;          /* 宣告指向整數的指標 ptri */
07     char *ptrc;         /* 宣告指向字元的指標 ptrc */
08
09     printf("sizeof(ptri)=%d\n",sizeof(ptri));
10     printf("sizeof(ptrc)=%d\n",sizeof(ptrc));
11     printf("sizeof(*ptri)=%d\n",sizeof(*ptri));
12     printf("sizeof(*ptrc)=%d\n",sizeof(*ptrc));
13
14     system("pause");
15     return 0;
16  }
```

```
/* prog10_4 OUTPUT---
sizeof(ptri)=4   } 指標變數皆佔了 4 個
sizeof(ptrc)=4   } 位元組
sizeof(*ptri)=4
sizeof(*ptrc)=1
----------------------*/
```

指標操作的練習 (1/2)

```
01  /* prog10_5, 指標的操作練習 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int a=5,b=10;
07     int *ptr1,*ptr2;
08     ptr1=&a;                /* 將ptr1設為a的位址 */
09     ptr2=&b;                /* 將ptr2設為b的位址 */
10     *ptr1=7;                /* 將ptr1指向的內容設為7 */
11     *ptr2=32;               /* 將ptr2指向的內容設為32 */
12     a=17;                   /* 設定a為17 */
13     ptr1=ptr2;              /* 設定ptr1=ptr2 */
14     *ptr1=9;                /* 將ptr1指向的內容設為9 */
15     ptr1=&a;                /* 將ptr1設為a的位址 */
16     a=64;                   /* 設定a為64 */
17     *ptr2=*ptr1+5;          /* 將ptr2指向的內容設為*ptr1+5*/
18     ptr2=&a;                /* 將ptr2設為a的位址 */
19     printf("a=%2d, b=%2d, *ptr1=%2d, *ptr2=%2d\n",a,b,*ptr1,*ptr2);
20     printf("ptr1=%p, ptr2=%p\n",ptr1,ptr2);
21     system("pause");
22     return 0;
23  }
```

```
/* prog10_5 OUTPUT------------
a=64, b=69, *ptr1=64, *ptr2=64
ptr1=0022FF6C, ptr2=0022FF6C
---------------------------------*/
```


指標操作的練習 (2/2)

執行6~18行時，變數變化的情形（&a=FF6C，&b= FF68）

行號	程式碼	a	b	ptr1	*ptr1	ptr2	*ptr2
06	int a=5,b=10;	5	10				
07	int *ptr1,*ptr2;	5	10	殘值	殘值	殘值	殘值
08	ptr1=&a;	5	10	FF6C	5	殘值	殘值
09	ptr2=&b;	5	10	FF6C	5	FF68	10
10	*ptr1=7;	7	10	FF6C	7	FF68	10
11	*ptr2=32;	7	32	FF6C	7	FF68	32
12	a=17;	17	32	FF6C	17	FF68	32
13	ptr1=ptr2;	17	32	FF68	32	FF68	32
14	*ptr1=9;	17	9	FF68	9	FF68	9
15	ptr1=&a;	17	9	FF6C	17	FF68	9
16	a=64;	64	9	FF6C	64	FF68	9
17	*ptr2=*ptr1+5;	64	69	FF6C	64	FF68	69
18	ptr2=&a;	64	69	FF6C	64	FF6C	64

當指標指向錯誤的型態時

- 指標指向不正確的型態所產生的錯誤：

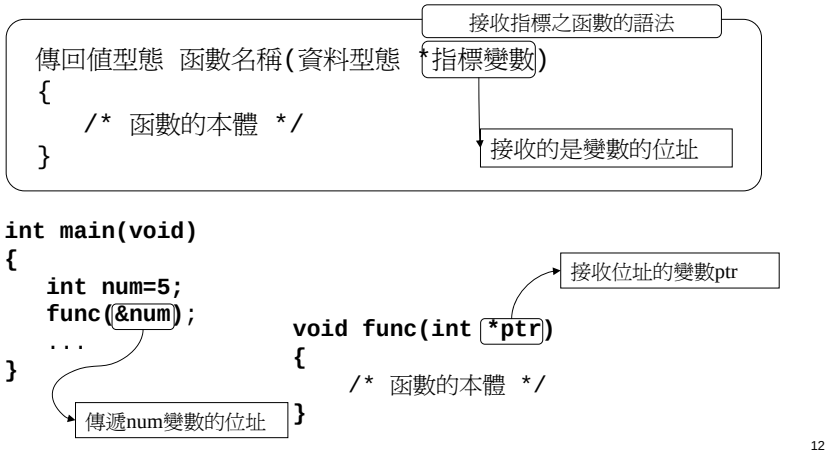
```
01  /* prog10_6, 錯誤的指標型態 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int a1=100, *ptri;
07     float a2=3.2f, *ptrf;
08     ptri=&a2;      /* 錯誤，將int型態的指標指向float型態的變數 */
09     ptrf=&a1;      /* 錯誤，將float型態的指標指向int型態的變數 */
10     printf("sizeof(a1)=%d\n",sizeof(a1));
11     printf("sizeof(a2)=%d\n",sizeof(a2));
12     printf("a1=%d,*ptri=%d\n",a1,*ptri);
13     printf("a2=%.1f,*ptrf=%.1f\n",a2,*ptrf);
14     system("pause");
15     return 0;
16  }
```

```
/* prog10_6 OUTPUT------
sizeof(a1)=4
sizeof(a2)=4
a1=100,*ptri=-1717986918
a2=3.2,*ptrf=0.0
-------------------------*/
```

incompatible pointer assignment:
dev C++ 4.9.9.2: warnings only
vc2010, dev C++ 5.8.3: errors

傳遞指標到函數 (1/3)

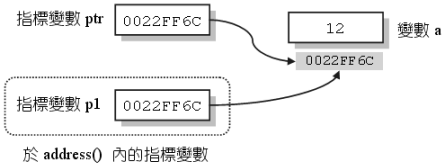
- 接收指標的函數：



傳遞指標到函數 (2/3)

- 傳遞指標到函數

```
/* prog10_7 OUTPUT-------------
於位址 0022FF6C 內，儲存的變數內容為 12
於位址 0022FF6C 內，儲存的變數內容為 12
--------------------------------*/
```

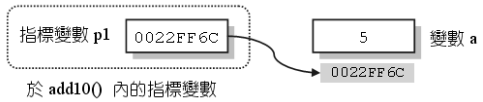



```
01  /* prog10_7, 傳遞指標到函數裡 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void address(int *);
05  int main(void)
06  {
07     int a=12;
08     int *ptr=&a;
09
10     address(&a);          /* 將 a 的位址傳入 address()函數中 */
11     address(ptr);         /* 將 ptr 傳入 address()函數中 */
12
13     system("pause");
14     return 0;
15  }
16  void address(int *p1)
17  {
18     printf("於位址%p 內，儲存的變數內容為%d\n",p1,*p1);
19  }
```


傳遞指標到函數 (3/3)

- 傳遞指標的應用：

```
/* prog10_8 OUTPUT---
呼叫 add10()之前, a=5
呼叫 add10()之後, a=15
----------------------*/
```




```
01  /* prog10_8, 傳遞指標的應用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void add10(int *);
05  int main(void)
06  {
07     int a=5;
08     printf("呼叫 add10()之前,a=%d\n",a);
09     add10(&a);              /* 呼叫 add10()函數 */
10     printf("呼叫 add10()之後,a=%d\n",a);
11     system("pause");
12     return 0;
13  }
14  void add10(int *p1)
15  {
16     *p1=*p1+10;     // 透過指標變數p1間接存取變數 a
17  }
```


變數值的互換 (錯誤)

```
/* prog10_9 OUTPUT---
交換前... a=5,b=20
交換後... a=5,b=20
--------------------*/
```

```
01  /* prog10_9, 將 a 與 b 值互換(錯誤示範) */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void swap(int,int);
05  int main(void)
06  {
07     int a=5,b=20;
08     printf("交換前... ");
09     printf("a=%d,b=%d\n",a,b);
10     swap(a,b);
11     printf("交換後... ");
12     printf("a=%d,b=%d\n",a,b);
13
14     system("pause");
15     return 0;
16  }
17
18  void swap(int x,int y)
19  {
20     int tmp=x;
21     x=y;
22     y=tmp;
23  }
```

a	b	x	y	tmp
5	20			
5	20	5	20	??
5	20	5	20	5
5	20	20	20	5
5	20	20	5	5

變數值的互換 (正確)

```
/* prog10_10 OUTPUT-
交換前... a=5,b=20
交換後... a=20,b=5
-------------------*/
```

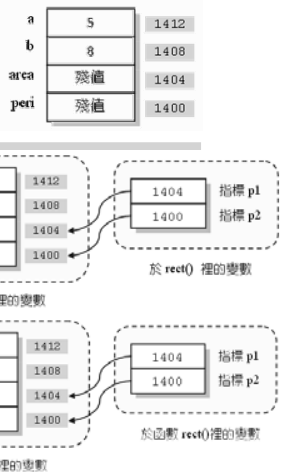
```
01  /* prog10_10, 將 a 與 b 值互換(正確範例) */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void swap(int *,int *);
05  int main(void)
06  {
07     int a=5,b=20;
08     printf("交換前... ");
09     printf("a=%d,b=%d\n",a,b);
10     swap(&a,&b);
11     printf("交換後... ");
12     printf("a=%d,b=%d\n",a,b);
13
14     system("pause");
15     return 0;
16  }
17
18  void swap(int *p1,int *p2)
19  {
20     int tmp=*p1;
21     *p1=*p2;
22     *p2=tmp;
23  }
```

a	b	p1	p2	tmp
5	20			
5	20	&a	&b	??
5	20	&a	&b	5
20	20	&a	&b	5
20	5	&a	&b	5

傳回多個數值的函數

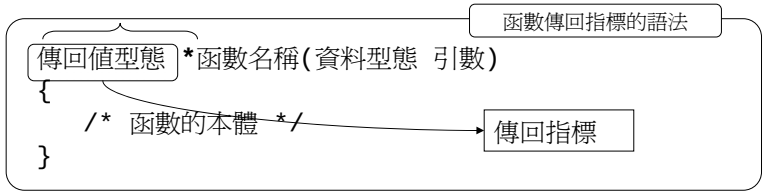
```
/* prog10_11 OUTPUT----
area=40,total length=26
----------------------*/
```

```
01  /* prog10_11, 傳回多個數值的函數 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void rect(int,int,int *,int *);
05  int main(void)
06  {
07     int a=5,b=8;
08     int area,peri;
09     rect(a,b,&area,&peri);
10     printf("area=%d,total length=%d\n"
11
12     system("pause");
13     return 0;
14  }
15
16  void rect(int x,int y,int *p1,int *p2)
17  {
18     *p1=x*y;
19     *p2=2*(x+y);
20  }
```



由函數傳回指標 (1/2)

- 傳回指標的函數：




```
int main(void)
{
   int *ptr,num;
   ptr=func(num);
   ...
}
```

接收函數所傳回的指標

傳回 指向整數的指標
(整數變數的記憶體位址)

```
int *func(int num)
{
    /* 函數的本體 */
    return 整數的指標;
    …
}
```

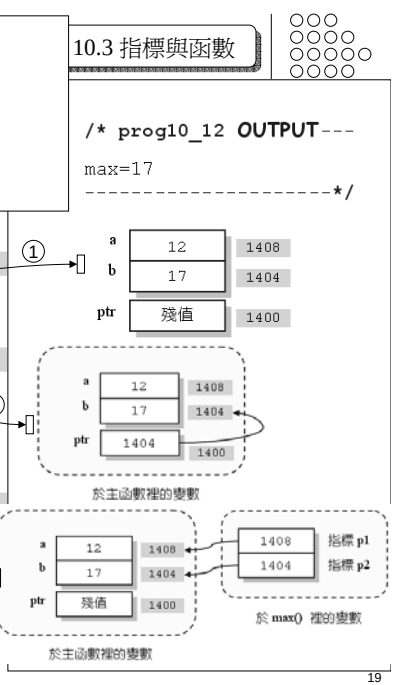

由函數

```
int *ptr = (int *) malloc(sizeof(int)*n);
…
return ptr;

int i:
for (i=0; i<n; i++)
    ptr[i] = i;  // *(ptr+i) = i;

if (ptr) free(ptr);
```

```
/* prog10_12 OUTPUT---
max=17
----------------------*/
```

```
01  /* prog
02  #include <stdio.h>
03  #include <stdlib.h>
04  int *max(int *,int *);
05  int main(void)
06  {
07     int a=12,b=17,*ptr;
08     ptr=max(&a,&b);
09     printf("max=%d\n",*ptr);
10
11     system("pause");
12     return 0;
13  }
14  int *max(int *p1, int *p2)
15  {
16     if(*p1>*p2)
17        return p1;
18     else
19        return p2;
20  }
```

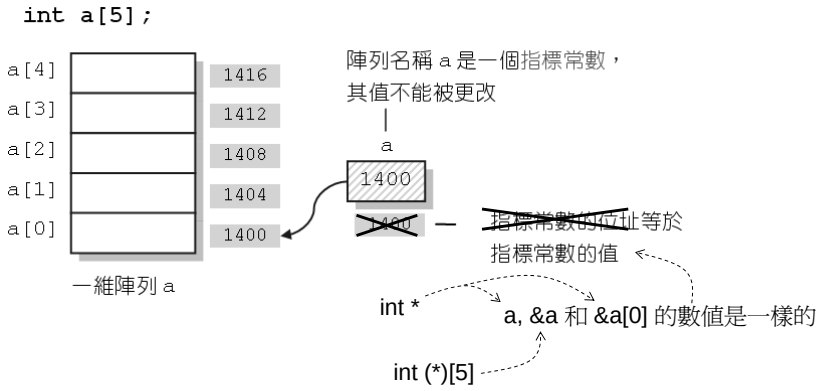
傳回 p1 與 p2 所指向之整數中，數值較大之整數的位址



指標與一維陣列

int *(const) a

- 陣列的名稱是一個指標常數，它指向該陣列的位址

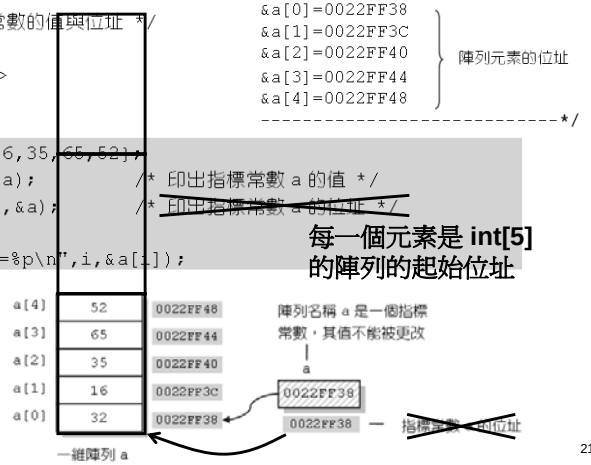


陣列名稱的值即陣列的位址

- 驗證陣列名稱是一個指標常數：

```
01  /* prog10_13, 指標常數的值與位址 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int i,a[5]={32,16,35,65,52};
07     printf("a=%p\n",a);          /* 印出指標常數 a 的值 */
08     printf("&a=%p\n",&a);        /* 印出指標常數 a 的位址 */
09     for(i=0;i<5;i++)
10       printf("&a[%d]=%p\n",i,&a[i]);
11     system("pause");
12     return 0;
13  }
```

```
/* prog10_13 OUTPUT----------
a=0022FF38      指標常數 a 的值
&a=0022FF38     指標常數 a 的位址
&a[0]=0022FF38
&a[1]=0022FF3C
&a[2]=0022FF40   陣列元素的位址
&a[3]=0022FF44
&a[4]=0022FF48
----------------------------*/
```



每一個元素是 int[5] 的陣列的起始位址

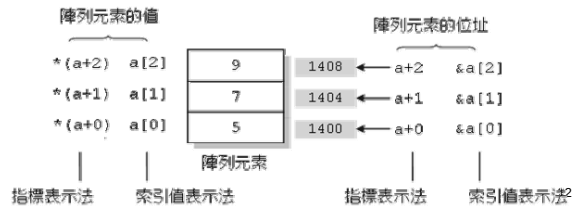
指標的算術運算 (1/3)

double *ptr;
ptr+5 時將存放在變數 ptr 裡的記憶體位址加上 5 * sizeof(double)

- 利用指標存取陣列的內容

```
01  /* prog10_14, 利用指標常數來存取陣列的內容 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int a[3]={5,7,9};
07     printf("a[0]=%d, *(a+0)=%d\n",a[0],*(a+0));
08     printf("a[1]=%d, *(a+1)=%d\n",a[1],*(a+1));
09     printf("a[2]=%d, *(a+2)=%d\n",a[2],*(a+2));
10     system("pause");
11     return 0;
12  }
```

```
/* prog10_14 OUTPUT--
a[0]=5, *(a+0)=5
a[1]=7, *(a+1)=7
a[2]=9, *(a+2)=9
------------------------*/
```



如果指標 a 指向某一個陣列，則 a+i 指向陣列裡索引值爲 i 的元素。

指標的算術運算 (2/3)

- 利用指標計算一維陣列內所有元素的和

```
01  /* prog10_15, 利用指標求陣列元素和 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int a[3]={5,7,9};
07     int i,sum=0;
08     for(i=0;i<3;i++)
09        sum+=*(a+i);        /* 加總陣列元素的總和 */
10     printf("sum=%d\n",sum);
11
12     system("pause");
13     return 0;
14  }
```

Funny thought:
a[i] 等效於 *(a+i)
*(a+i) 等效於 *(i+a)
那麼 i[a] 是什麼東西?

```
/* prog10_15 OUTPUT--
sum=21
------------------------*/
```


指標的算術運算 (3/3)

- 改以指標變數 ptr 來指向陣列 a，並計算總和：

```
01  /* prog10_16, 利用指標求陣列元素和 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int a[3]={5,7,9};
07     int i,sum=0;
08     int *ptr=a;             /* 設定指標 ptr 指向陣列元素 a[0] */
09     for(i=0;i<3;i++)
10        sum+=*(ptr++);       /* 計算陣列元素值的累加  */
11     printf("sum=%d\n",sum);
12
13     system("pause");
14     return 0;
15  }
```

Very commonly seen in legacy codes like UNIX source. But, definitely not recommended for readability and maintainability.

for (i=0; i<3; i++)
sum+=a[i];

for (i=0; i<3; i++)
sum+=*ptr++;

~~for (i=0; i<3; i++)~~
~~sum+=(*ptr)++;~~

```
/* prog10_16 OUTPUT--
sum=21
------------------------*/
```

傳遞一維陣列到函數裡

```
01  /* prog10_17, 將陣列第 n 個元素的值取代為 num */
02  #include<stdio.h>
03  #include <stdlib.h>
04  void replace(int *,int,int);   /* 宣告 replac
05  int main(void)
06  {
07     int a[5]={13,32,67,14,95};
08     int i,num=24;
09
10     replace(a,4,num);           /* 呼叫函數 replace() */
11     printf("置換後，陣列的內容為");
12     for(i=0;i<5;i++)            /* 印出陣列的內容 */
13        printf("%3d",a[i]);
14     printf("\n");
15     system("pause");
16     return 0;
17  }
18  void replace(int *ptr,int n,int num)
19  {
20     *(ptr+n-1)=num;      /* 將陣列第 n 個元素設值為 num */
21  }
```

```
/* prog10_17 OUTPUT------------
置換後，陣列的內容為 13 32 67 24 95
-------------------------------*/
```

```
void replace(int a[5])
void replace(int a[])
void replace(int * const a)
void replace(int *ptr) {
   …
   int b[20];
   ptr = b;
   for (int i=0; i<20; i++)
      ptr[i] = 10;
}
```

ptr[n-1]=num;

函數傳回指標

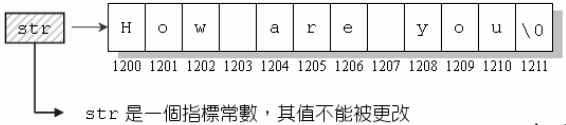
```
01  /* prog10_18, 函數傳回值為指標 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define SIZE 5
05  int *maximum(int *);       /* 宣告 maximum()函數的原型 */
06  int main(void)
07  {
08     int a[SIZE]={3,1,7,2,6};
09     int i,*ptr;
10     printf("array a=");
11     for(i=0;i<SIZE;i++)
12        printf("%d ",a[i]);
13     ptr=maximum(a);          /* 呼叫 maximum()函數，並傳入陣列 a */
14     printf("\nmaximum=%d\n",*ptr);
16     system("pause");
17     return 0;
18  }
19  int *maximum(int *arr)     /* 定義 maximum()函數 */
20  {
21     int i,*max;
22     max=arr;                /* 設定指標 max 指向陣列的第一個元素 */
23     for(i=1;i<SIZE;i++)
24        if(*max < *(arr+i))
25           max=arr+i;
26     return max;             /* 傳回最大值之元素的位址 */
27  }
```

```
/* prog10_18 OUTPUT--
array a=3 1 7 2 6
maximum=7
-----------------------*/
```


以指標變數指向字串 (1/2)

- 利用字元陣列來儲存字串：

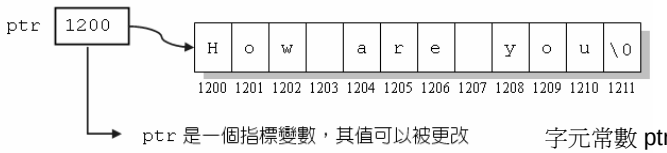
```
char str[]="How are you";
```



字元變數 str[i] 可以更改

- 利用指標指向字串：

```
const char *ptr="How are you";
```



字元常數 ptr[i] 不可以更改

以指標變數指向字串 (2/2)

- 以指標變數指向字串的範例：

```
01  /*  prog10_19, 以指標變數指向字串 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     char name[20];
07     char *ptr="How are you?";     /* 將指標指向字串"How are you?" */
08     printf("what's your name? ");
09     gets(name);                   /* 由鍵盤讀入字串 */
10     printf("Hi, %s, ",name);      /* 印出字串陣列 name 的內容 */
11     puts(ptr);                    /* 印出由 ptr 所指向的字串 */
12
13     system("pause");
14     return 0;
15  }
```

```
/* prog10_19 OUTPUT---
what's your name? Wien
Hi, Wien, How are you?
-----------------------*/
```

指標陣列 (1/3)

- 一維指標陣列的宣告格式：

宣告指標陣列

```
資料型態 *陣列名稱[元素個數];
```

- 以二維的字元陣列來儲存字串陣列：

```
char str[3][10]={"Tom", "Lily", "James Lee"};
```

浪費掉的空間

- 以指標陣列的方式來撰寫：

```
char *ptr[3]={"Tom", "Lily", "James Lee"};
```

空間不浪費

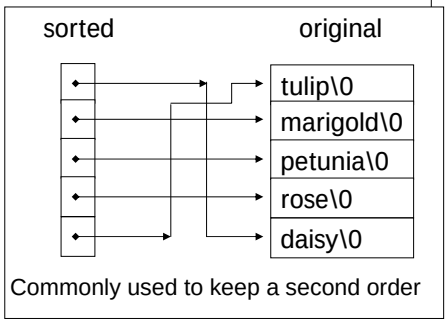
指標陣列 (2/3)

- 指標陣列的範例：

```
/* prog10_20 OUTPUT--

Tom
Lily
James Lee
----------------------*/
```

```
01  /* prog10_20, 指標陣列 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int i;
07     char *ptr[3]={"Tom", "Lily", "James Lee"};
08     for(i=0;i<3;i++)
09        puts(ptr[i]);   /* 印出指標 ptr[i]所指向的字串 */
10
11     system("pause");
12     return 0;
13  }
```


指標陣列 (3/3)



Commonly used to keep a second order

```
#include <stdlib.h> // qsort
#include <string.h> // strcmp

char flowers[][20] = {"tulip", "marigold", "petunia", "rose", "daisy"};
qsort(flowers, 5, 20, strcmp);

----------------------------------------------------------

char *flowerSorted[5];
for (i=0; i<5; i++)
   flowerSorted[i] = &flowers[i][0]; // flowers[i]
qsort(flowersSorted, 5, sizeof(char *), strcmp);
```


指向指標的指標 (1/3)

- 雙重指標存放指標變數的位址

雙重指標宣告的格式

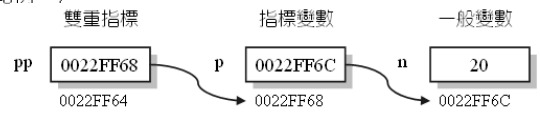
```
資料型態 **雙重指標;
```

- 宣告雙重指標的範例：
  - int **ptri;
  - char **ptrc;

指向指標的指標 (2/3)

- 雙重指標的範例：

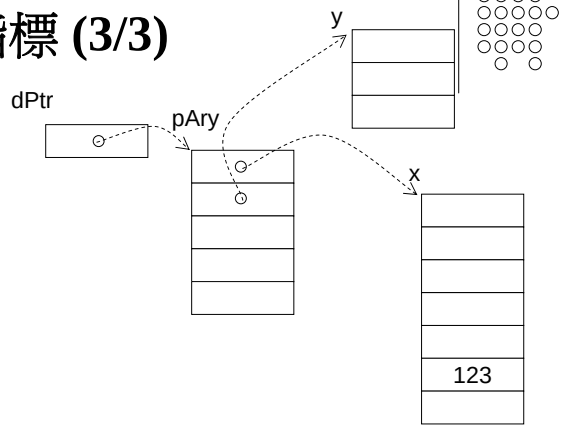
```
01  /* prog10_21, 雙重指標的範例 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int n=20,*p,**pp;
07     p=&n;
08     pp=&p;
09     printf("n=%d,&n=%p,*p=%d,p=%p,&p=%p\n",n,&n,*p,p,&p);
10     printf("**pp=%d,*pp=%p,pp=%p,&pp=%p\n",**pp,*pp,pp,&pp);
11
12     system("pause");
13     return 0;
14  }
```

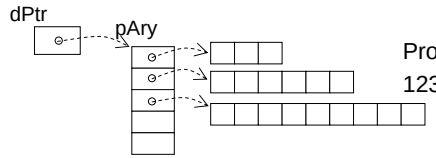



```
/* prog10_21 OUTPUT------------------------

n=20,&n=0022FF6C,*p=20,p=0022FF6C,&p=0022FF68
**pp=20,*pp=0022FF6C,pp=0022FF68,&pp=0022FF64
-------------------------------------------*/
```


指向指標的指標 (3/3)

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 int main(void) {
04     int *pAry[5];
05     int **dPtr = pAry;
06     int x[7], y[3];
07     pAry[0] = x;
08     pAry[1] = y;
09     x[5] = 123;
10     printf("%d %d %d\n", x[5], pAry[0][5], dPtr[0][5]);
11     system("pause");
12     return 0;
13 }
```



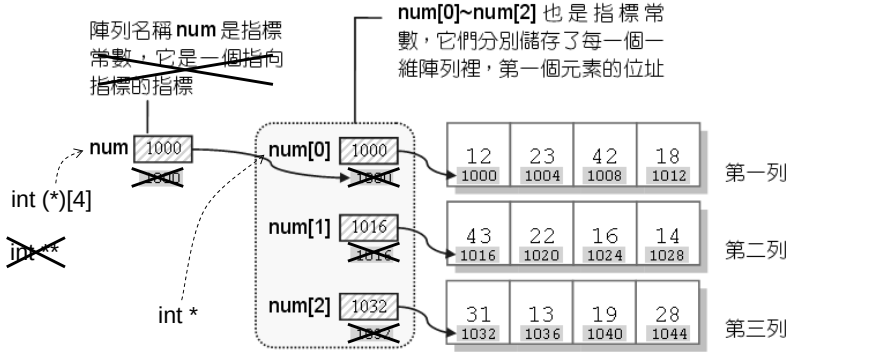


Program output
123 123 123

二維陣列與雙重指標的關係

int num[3][4]

- 由 3 個一維陣列所組成，每個一維陣列裡各有4個整數
- num[0]、num[1] 與 num[2] 爲整數指標常數 int *const，它們分別指向 這3個一維陣列



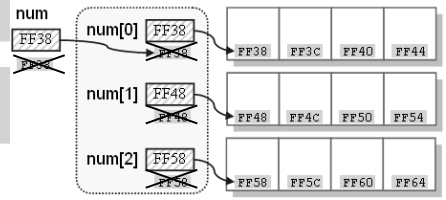
驗證二維陣列與指標的關係

```
01  /* prog10_22, 印出陣列的位址 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int num[3][4];
07
08     printf("num=%p\n",num);
09     printf("&num=%p\n",&num);
10     printf("*num=%p\n",*num);
11
12     printf("num[0]=%p\n",num[0]);
13     printf("num[1]=%p\n",num[1]);
14     printf("num[2]=%p\n",num[2]);
15
16     printf("&num[0]=%p\n",&num[0]);
17     printf("&num[1]=%p\n",&num[1]);
18     printf("&num[2]=%p\n",&num[2]);
19     system("pause");
20     return 0;
21  }
```

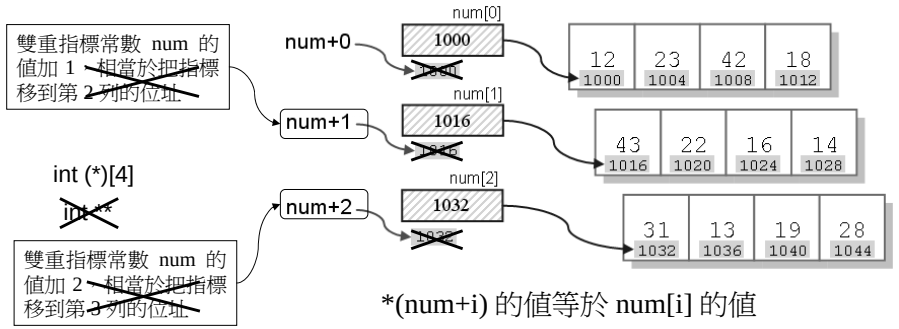
```
/* prog10_22 OUTPUT--
num=0022FF38
&num=0022FF38
*num=0022FF38
num[0]=0022FF38
num[1]=0022FF48
num[2]=0022FF58
&num[0]=0022FF38
&num[1]=0022FF48
&num[2]=0022FF58
----------------------*/
```

指標常數的值

指標常數的位址



二維陣列的指標表示方式

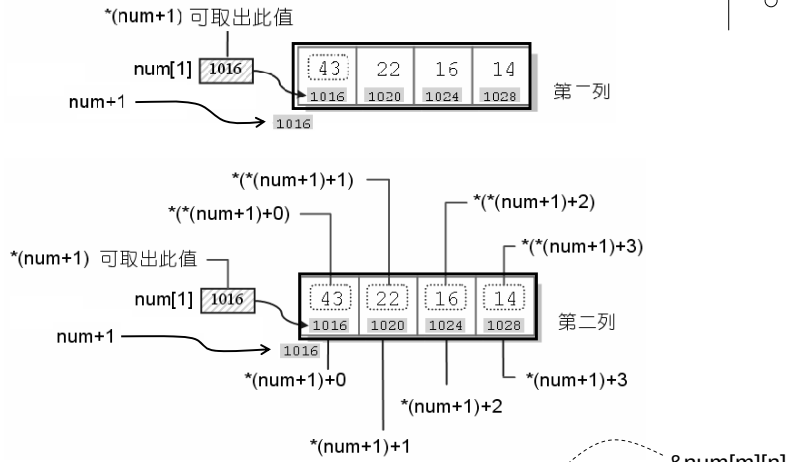


*(num+i) 的值等於 num[i] 的值

如果 num 的型態是 int **
則 num+1 時指標只會加 sizeof(int*)
因爲 num 的型態是 int (*)[4]
所以 num+1 時指標會加 sizeof(int [4])

```
int num[3][4];
fun(num);
…
void fun(int (*y)[4]) { … }
void fun(int **y) { … }
```


取得每一列裡特定的元素



- 取出第 m+1 列，第 n+1 行的位址：*(num+m)+n
- 取出第 m+1 列，第 n+1 行的元素： *(*(num+m)+n)

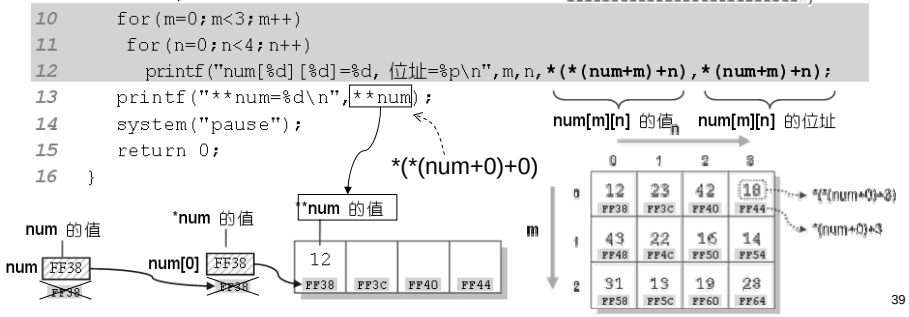
&num[m][n]

num[m][n]

以指標操作二維陣列 (1/2)

```
01  /* prog10_23, 印出陣列的位址 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int num[3][4]={{12,23,42,18},
07                    {43,22,16,14},
08                    {31,13,19,28}};
09     int m,n;
10     for(m=0;m<3;m++)
11      for(n=0;n<4;n++)
12        printf("num[%d][%d]=%d, 位址=%p\n",m,n,*(*(num+m)+n),*(num+m)+n);
13     printf("**num=%d\n",**num);
14     system("pause");
15     return 0;
16  }
```

```
--------prog10_23 OUTPUT--------
num[0][0]=12, 位址=0022FF38
num[0][1]=23, 位址=0022FF3C
num[0][2]=42, 位址=0022FF40
num[0][3]=18, 位址=0022FF44
num[1][0]=43, 位址=0022FF48
num[1][1]=22, 位址=0022FF4C
num[1][2]=16, 位址=0022FF50
num[1][3]=14, 位址=0022FF54
num[2][0]=31, 位址=0022FF58
num[2][1]=13, 位址=0022FF5C
num[2][2]=19, 位址=0022FF60
num[2][3]=28, 位址=0022FF64
**num=12
--------------------------*/
```



以指標操作二維陣列 (2/2)

```
01  /* prog10_24, 利用指標將大於 40 的陣列元素設值為 40 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int num[3][4]={{12,23,42,18},
07                    {43,22,16,14},
08                    {31,13,19,28}};
09     int m,n;
10     for(m=0;m<3;m++)
11     {
12        for(n=0;n<4;n++)
13        {
14           if(*(*(num+m)+n)>40)   /* 判別 num[m][n]的值是否大於 40 */
15              *(*(num+m)+n)=40;   /* 如果是，則將元素值設為 40 */
16           printf("%3d",*(*(num+m)+n));  /* 印出元素 num[m][n]的值 */
17        }
18        printf("\n");
19     }
20     system("pause");
21     return 0;
22  }
```

```
/* prog10_24 OUTPUT---
 12 23 40 18
 40 22 16 14
 31 13 19 28
----------------------*/
```

A last word:
Now that you know num[m][n] is equivalent to *(*(num+m)+n). Do not use this latter form in your program!!