

Chapter 10
Pointer



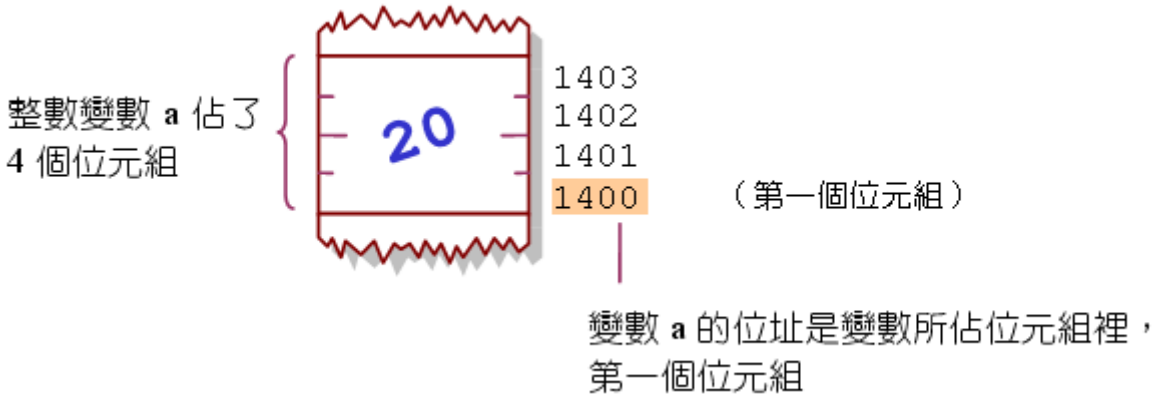
Outline

- Pointer
- Call by reference
- Pointers and arrays

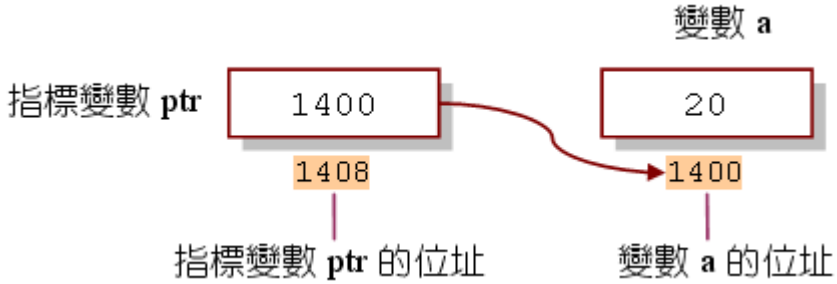


Address

- The address of a variable is the memory position of the first byte of that variable.



- Pointer variables are variables used to store the memory address.



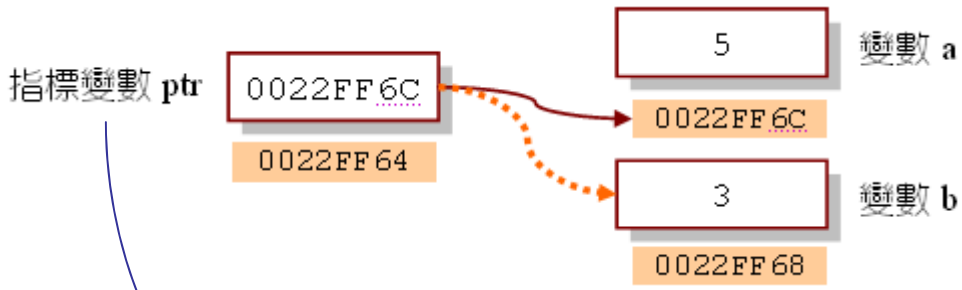
Why Pointers?

- Pointers could make parameter passing of functions more efficient.

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

Pass address of arrays, instead of the content of whole array

- Pointers could change the pointed address to maintain its flexibility.
  - Point to different variables.
  - Point to different functions to invoke different function calls.

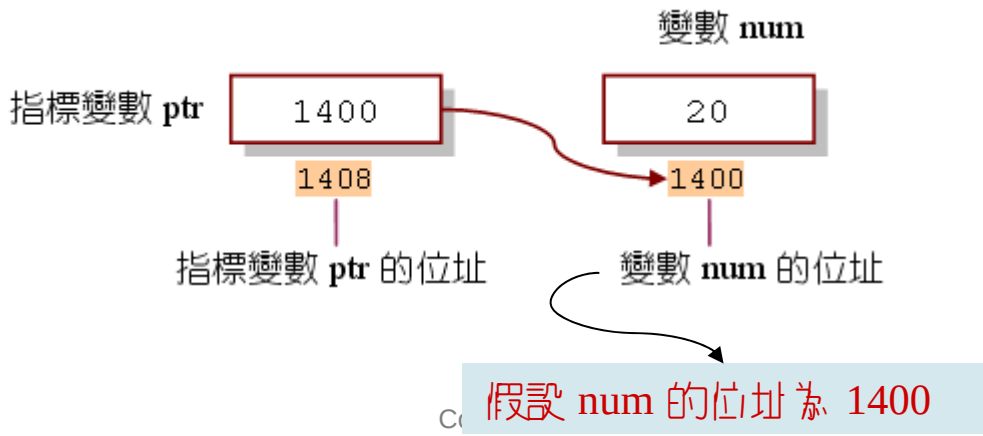




Pointer Definitions

- * used with pointer variables
  - int *ptr; // means that ptr is a pointer to an integer var
- Multiple pointers require a single * before each variable definition
  - int *ptr1, *ptr2; // declare two pointer variables (ptr1, ptr2)
  - Int ptr1, *ptr2; /* declare an integer variable (ptr1) and a pointer variable (ptr2) */
- ***Example***

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





Initialization

- Set a pointer variable to 0 or NULL
  - Point to nothing (NULL preferred)
  - int *ptr = 0;
  - double *ptr2 = NULL;



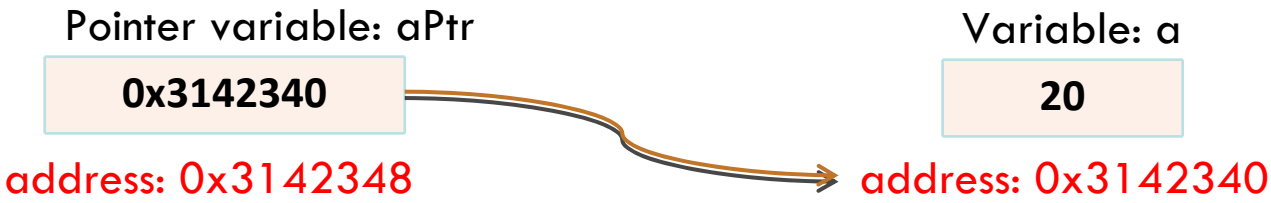
Initialization

- Set a pointer variable to 0 or NULL
- Point to nothing (NULL preferred)
  - int *ptr = 0;
  - double *ptr2 = **NULL**;

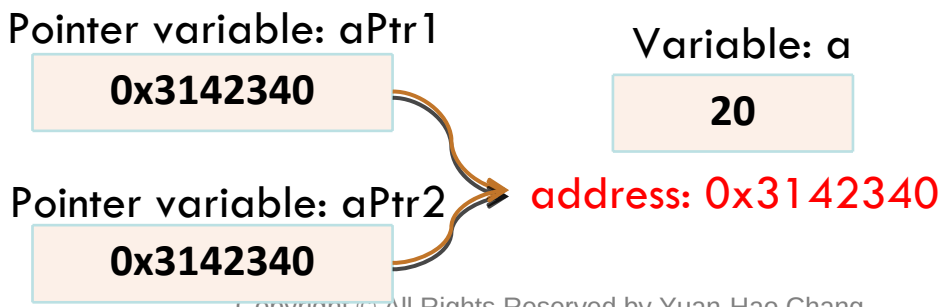


Initialization (Cont.)

- Set a pointer variable to an address
  - int *ptr = 0x3142340 // point to address 0x3142340
  - int a = 20;
    int *aPtr = &a; // point to the address of variable a
  - &: the operator used to get the address of a variable



  - int a = 20, *aPtr1, *aPtr2;
    aPtr1 = &a;
    aPtr2 = aPtr1;

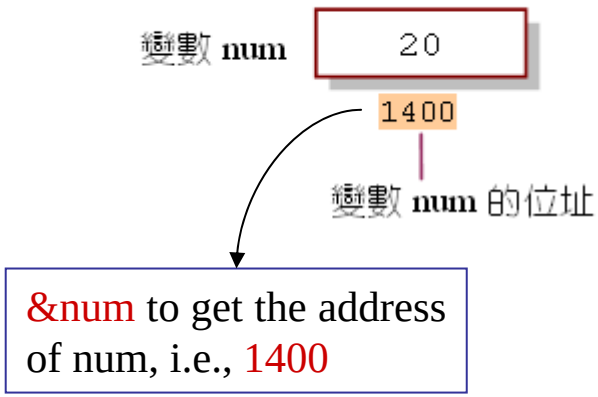


How to Use Pointers

- Two types of actions
  - Use a pointer to store the address of a variable
  - Use a pointer to access the pointed variable

Address operator「&」

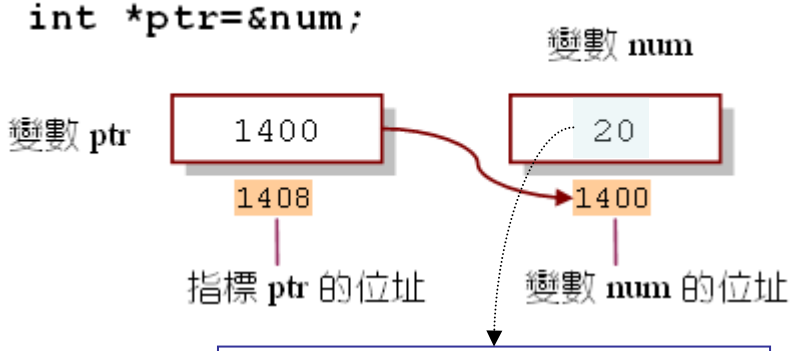
```
int num=20;
```



&num to get the address of num, i.e., 1400

Pointer operator「*」

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



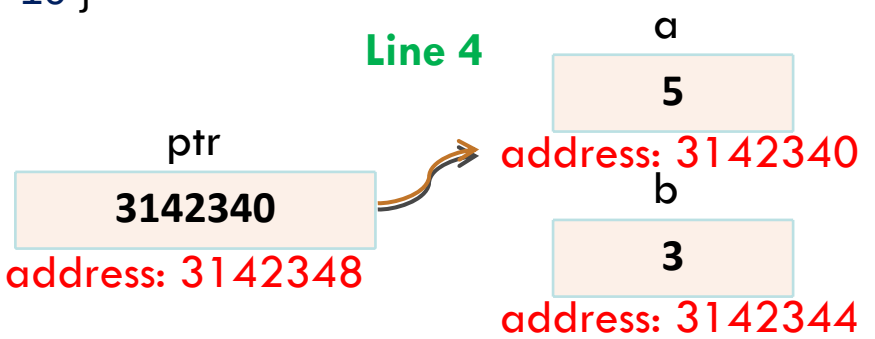
*ptr to get the content of num pointend by ptr, i.e., 20

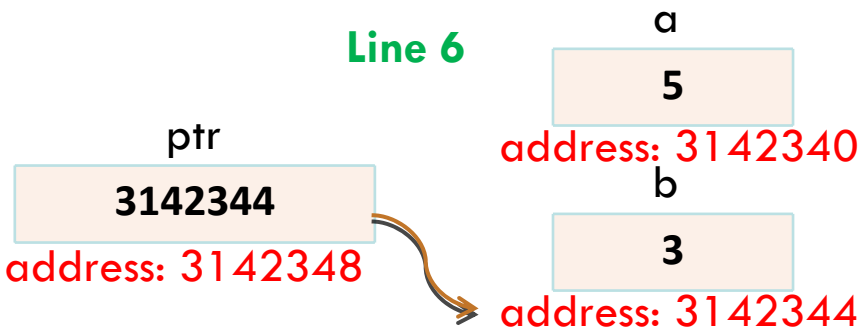


Example

```
01 int main() {}
02 int a = 5, b = 3;
03 int *ptr;
04 ptr = &a;
05 printf("&a=%p,&ptr=%p,ptr=%p,*ptr=%d\n", &a, &ptr, ptr, *ptr);
06 ptr = &b;
07 printf("&b=%p,&ptr=%p,ptr=%p,*ptr=%d\n", &b, &ptr, ptr, *ptr);
08 system("pause");
09 return 0;
10 }
```

&a=3142340, &ptr=3142348, ptr=3142340, *ptr=5
&b=3142344, &ptr=3142348, ptr=3142344, *ptr=3







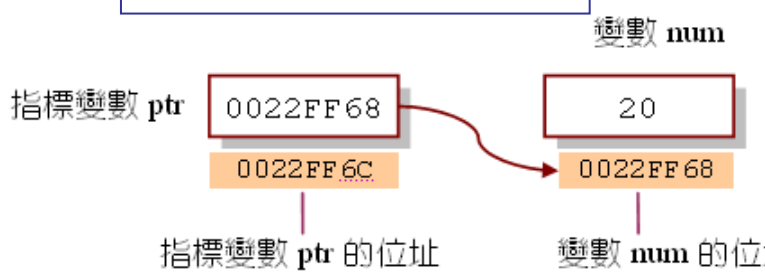
Another Example

```
01  /* 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  }
```

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

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

After execution of Line 6

After execution of Line 8



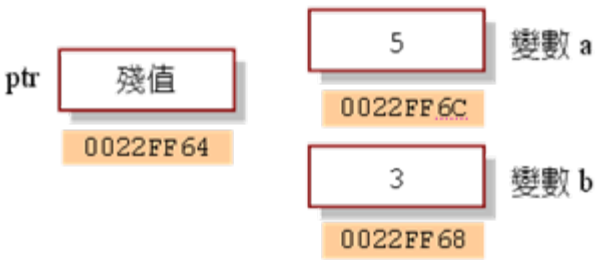


Another Example (Cont.)

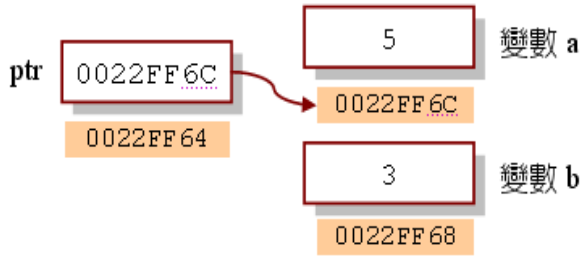
```c
/* 指標變數的使用 */
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   int a=5,b=3;
   int *ptr;

   ptr=&a;                /* 將 a 的位址設給指標 ptr 存放 */
   printf("&a=%p, &ptr=%p, ptr=%p, *ptr=%d\n",&a,&ptr,ptr,*ptr);
   ptr=&b;                /* 將 b 的位址設給指標 ptr 存放 */
   printf("&b=%p, &ptr=%p, ptr=%p, *ptr=%d\n",&b,&ptr,ptr,*ptr);

   system("pause");
   return 0;
}
```

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

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

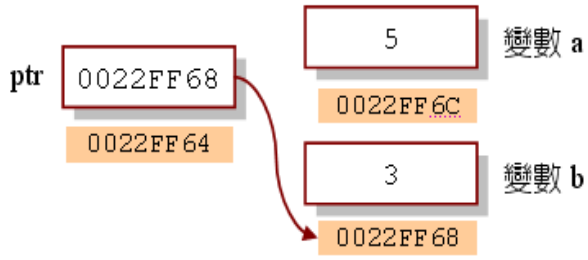
執行完第 7 行



執行完第 9 行



執行完第 11 行





Size of Pointer Variables

- The size of a pointer variable equals the size of an integer.

```
int main() {}
    int *ptri;
    double *ptrc;
    printf("sizeof(ptri) = %d\n", sizeof(ptri));
    printf("sizeof(ptrc) = %d\n", sizeof(ptrc));
    printf("sizeof(*ptri) = %d\n", sizeof(*ptri));
    printf("sizeof(*ptrc) = %d\n", sizeof(*ptrc));
    return 0;
}
```

output

```
sizeof(ptri) = 4
sizeof(ptrc) = 4
sizeof(*ptri) = 4
sizeof(*ptrc) = 8
```



Size of Pointer Variables (Cont.)

```
01  /* 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  }
```

```
/* 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
----------------------*/
```



Practice of Pointers

```
/* prog10_5 OUTPUT-----------

a=64, b=69, *ptr1=64, *ptr2=64
ptr1=0022FF6C, ptr2=0022FF6C

--------------------------------*/
```

```
01  /* 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  }
```



Practice of Pointers (Cont.)

Execution of Lines 6~18（&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



Semantic Error

- Point to an incorrect data type

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

```
01  /* 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  }
```



Semantic Error (Cont.)

- Access a null pointer or uninitialized pointer

```
int main() {}
    int *ptri;
    float *ptrf;
    printf("*ptri=%d\n", *ptri);
    printf("*ptrf=%f\n", *ptrf);
    return 0;
}
```

Program will be terminated



Function Call

Pointer as function parameter

```
ReturnType FuncName(DataType *Ptr)
{
    /* 函數的本體 */
}
```

Receive address of variables

```
int main(void)
{
    int num=5;
    func(&num);
    ...
}
```

Pass variable address

Receive variable address

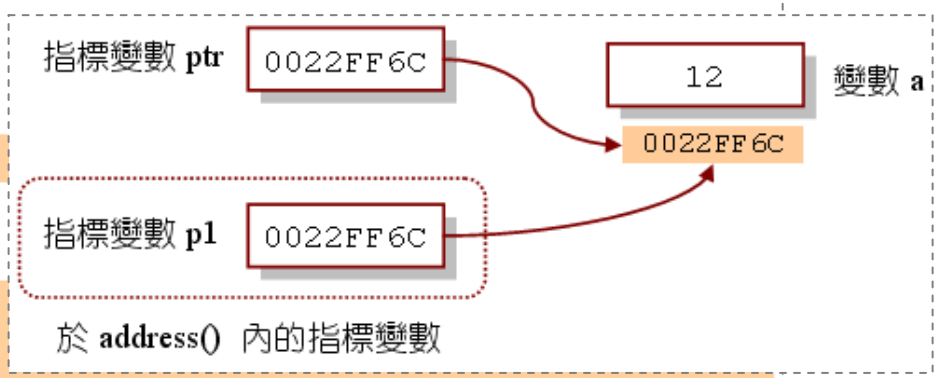
```
void func(int *ptr)
{
    /* 函數的本體 */
}
```



Example

```
/* 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
-------------------------------
```

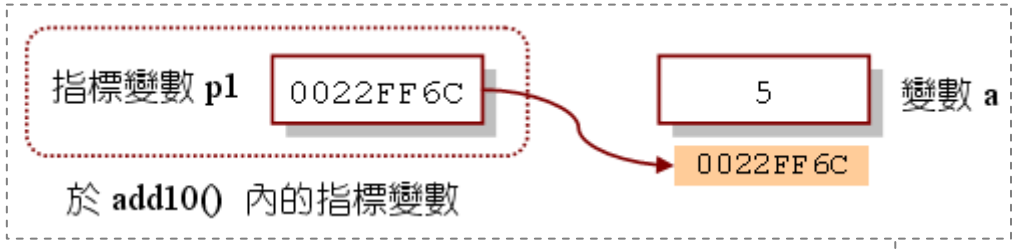
```
01  /* 傳遞指標到函數裡 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void address(int *);
05  int main(void)
06  {
07     int a=12;
08     int *ptr=&a;
09
10     address(&a);              /* 將 a 的位址傳入 address()函數中 */
11     address(ptr);             /* 將 ptr 傳入 address()函數中 */
12
13     system("pause");
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  }
```





Example (Cont.)

```
/* 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;
17  }
```

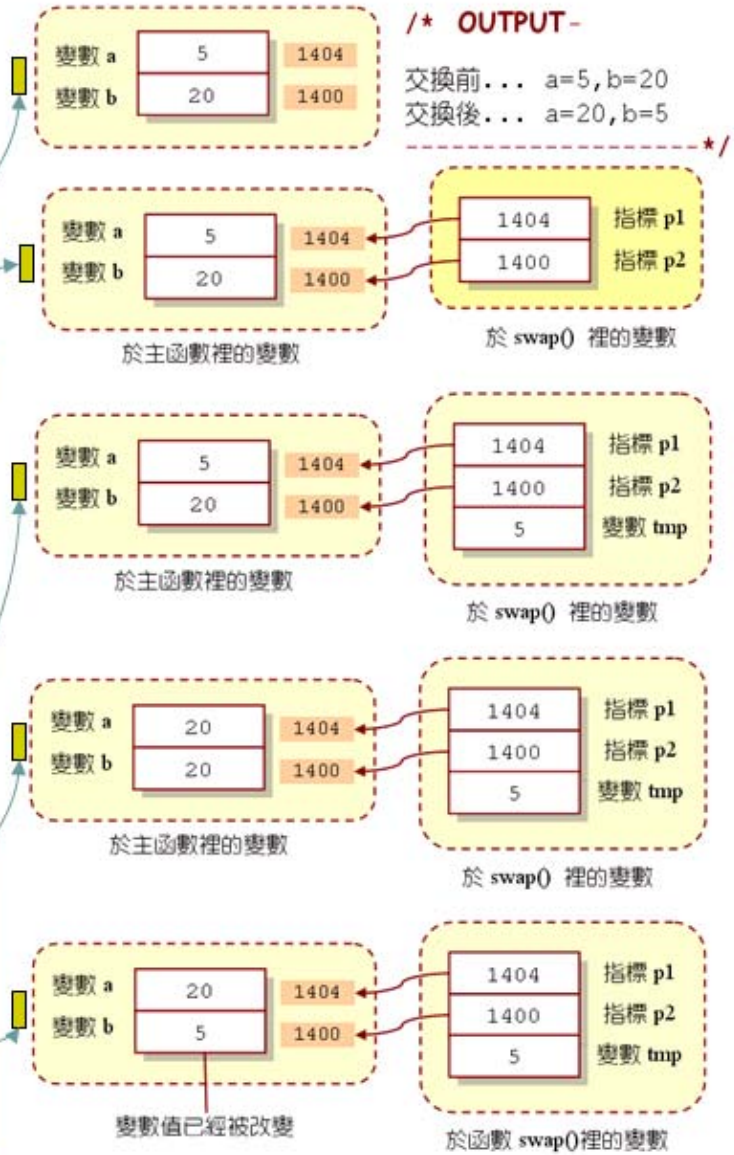
Wrong Example

```
01  /* 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  }
```

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

Correct Example

```
01  /* 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  }
```

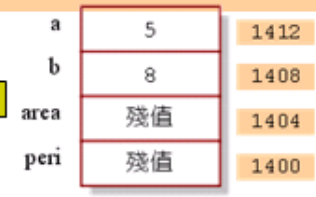


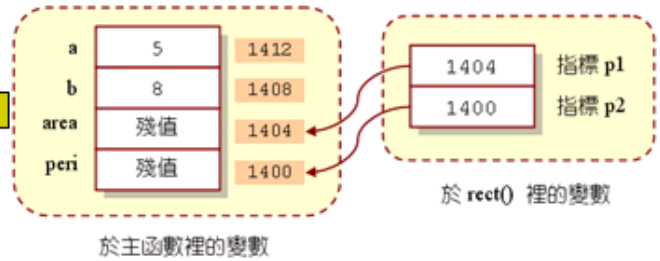


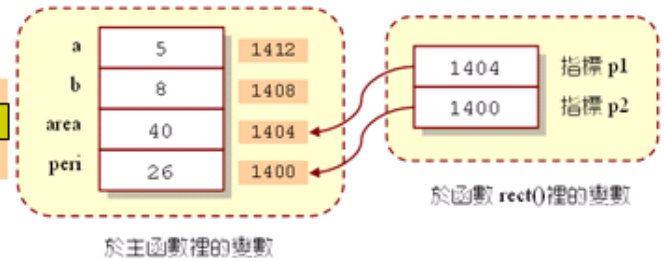
Multiple Pointer Parameters

```
01  /* 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",area,peri);
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  }
```

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









Pointers as Return Type

- Functions return pointers.

Pointers as Return Type

```
ReturnType *FuncName(DataType Para1)
{
    /* Body of function */
}
```

Return a pointer

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

Receive a pointer returned by func()

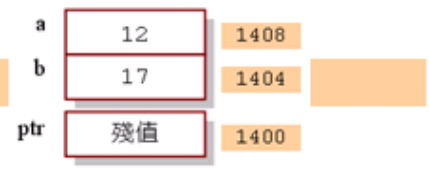
Return a pointer pointing to an "int" variable

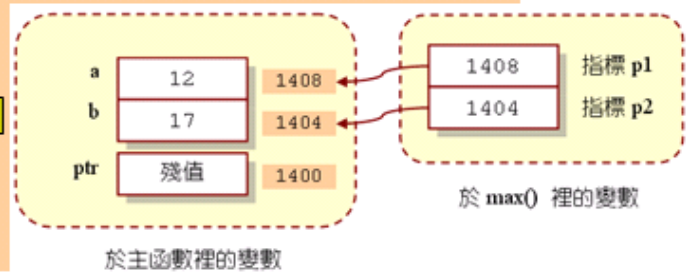
```
int *func(int num)
{
    /* Body of function */
}
```

Pointers as Return Type (Cont)

```
/*  OUTPUT---
max=17
--------------------*/
/* 由函數傳回指標  */
#include <stdio.h>
#include <stdlib.h>
int *max(int *,int *);
int main(void)
{
   int a=12,b=17,*ptr;
   ptr=max(&a,&b);
   printf("max=%d\n",*ptr);

   system("pause");
   return 0;
}
int *max(int *p1, int *p2)
{
   if(*p1>*p2)
      return p1;
   else
      return p2;
}
```

傳回 p1 與 p2 所指向之整數中，數值較大之整數的位址

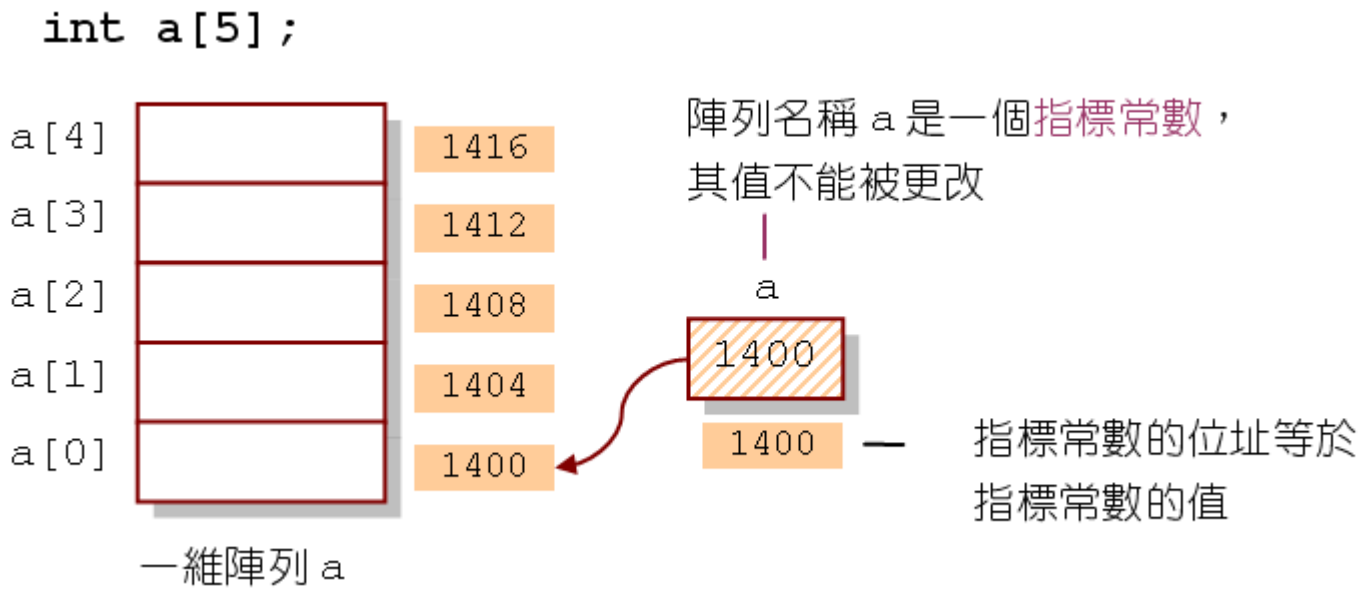






Pointer and Array

- Array's name is a ***pointer constant*** that points to the first address of that array.

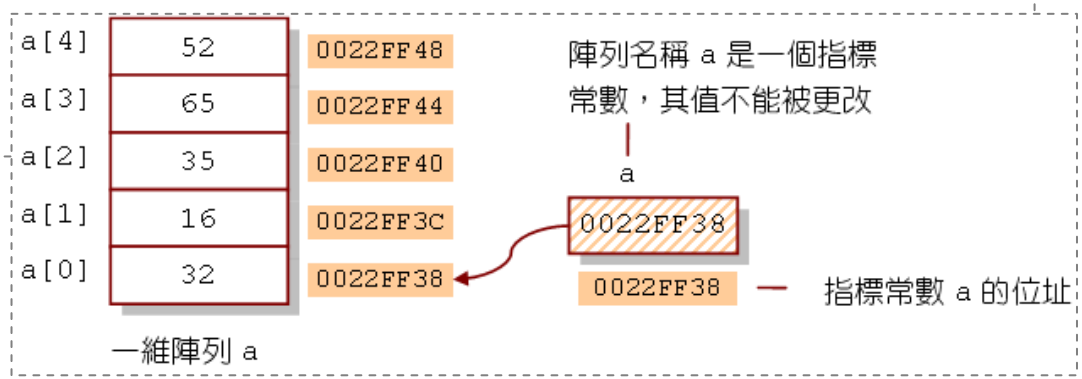




Pointer and Array (Cont.)

```
01  /* 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  }
```

```
/* 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
----------------------------*/
```



指標的位址等於它本身所存放的位址。

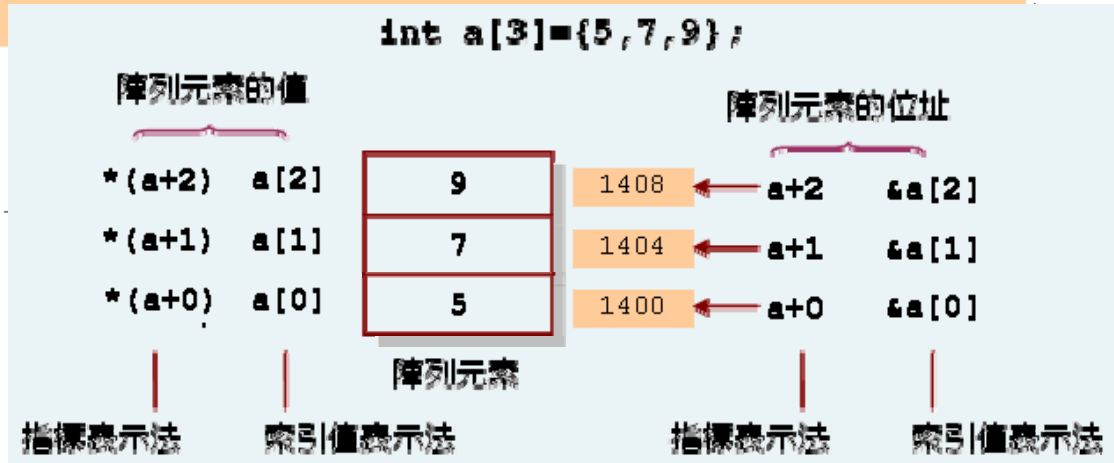


Array Access with Pointers

Pointer's address is pointer's value

```
01  /* 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, *
10     system("pause");
11     return 0;
12  }
```

```
/* 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，索引值為 i 的元素。



Array Access with Pointers (Cont.)

- Use pointer to sum up the values in the array.

```
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  }
```

```
/*  OUTPUT--

sum=21
-----------------------*/
```



Array Access with Pointers (Cont.)

```
01  /* 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  }
```

An array's name is a constant and can't be changed.

```
/* OUTPUT--

sum=21
-----------------------*/
```



Array and Function

- Pass an array to a function

```
01  /*  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 replace()函數的原型 */
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  }
```

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



Array and Function (Cont.)

- Return pointers

```
01  /* 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  }
```

```
/* OUTPUT--

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



Lab 10

- 假設整數陣列 arr 宣告為 int arr[5] = {1, 2, 3, 4, 5}。試撰寫一函數void square(int *arr)，在呼叫square()函數後，一維陣列arr裡的每一個元素皆會被平方，並印出平方後的結果。
- 假設整數陣列 arr 宣告為 int arr[5] = {31,17,33,22,16 }。試寫一個函數int max(int *arr) 以回傳陣列中的最大值與一個函數 int min(int *arr) 以回傳陣列中的最小值。並將 max() 與 min() 的回傳值顯示在螢幕上。