Chapter 9.
Pointers and String

I-Fen Chao

樂透號碼

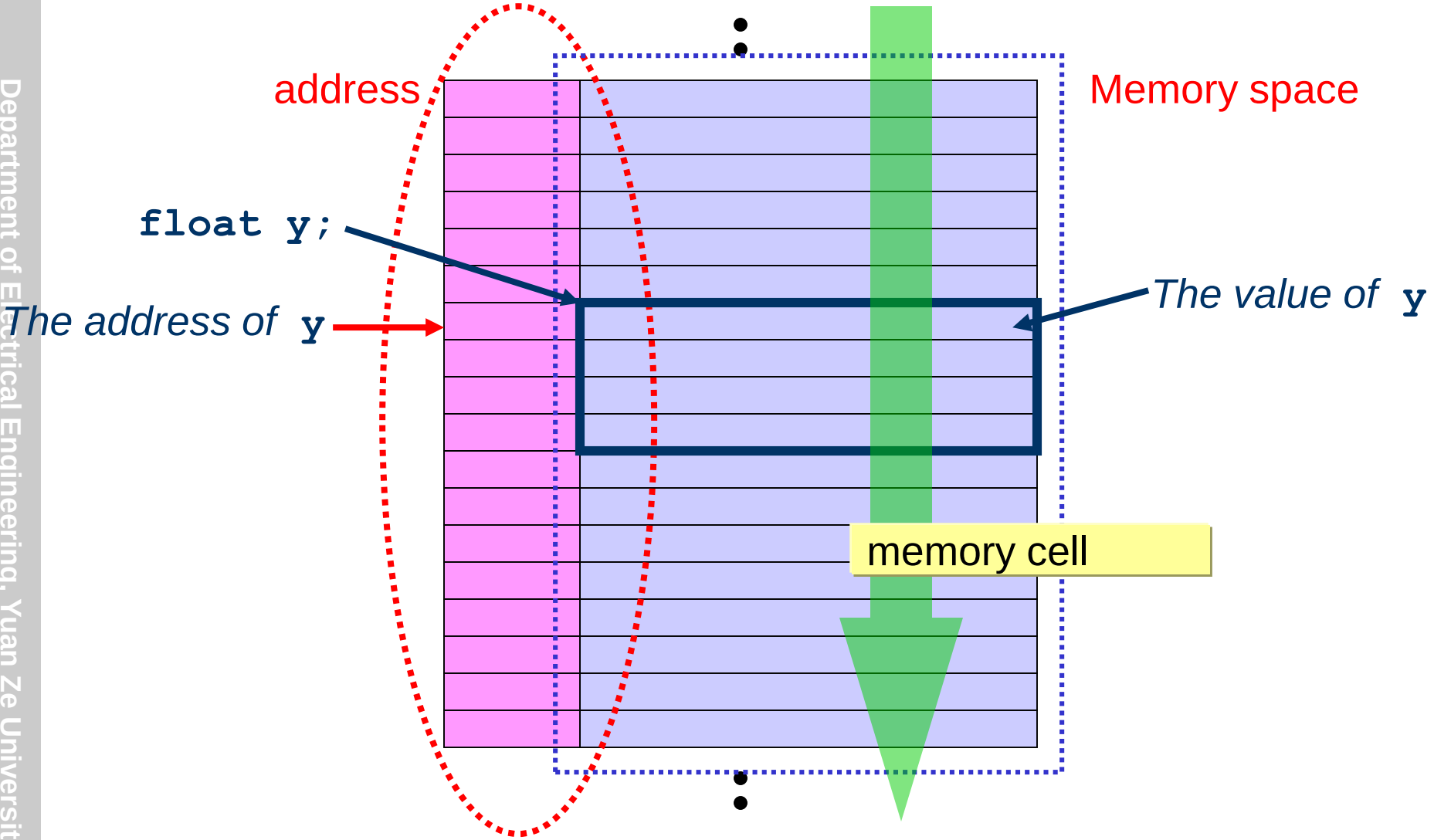
我買的號碼:
號碼1: xx
號碼2: xx
號碼3: xx
號碼4: xx
號碼5: xx
號碼6: xx
電腦樂透號碼: xx, xx, xx, xx, xx, xx
我中的號碼: ??
共中 x 個號碼

=

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

bool check_valid(int a[], int target);

void main(void)
{
    int user[6] = { 0 };
    int lot[6] = { 0 };//computer generate
    int i;
    int num;
    srand(time(NULL));

    //computer generation
    for (i = 0; i < 6; i++) {


    }

    //print lottery
    for (i = 0; i < 6; i++)
        printf("%d ", lot[i]);
    printf("\n");
```

```
28      //I buy:
29      printf("我買的號碼:\n");
30      for (i = 0; i < 6; i++) {
31          printf("號碼%d:", i+1);
32          scanf_s("%d", &num);
33
34
35
36
37
38
39      }
40
41      //check lottery
42      int j;
43      int count = 0;
44      for
45
46
47
48
49
50
51      }
52      printf("共中%d個號碼\n", count);
```

Variables and values stored in memory



Pointer variables

- The contents of this kind of variables were addresses.
- How to declare a pointer variables：

```
data_type *r_name;
```

r_name is a variable which stores an address

The way variables are stored in memory

```
int a,b;
float r,s;
int *c;
float *t;
```

Variable name	Variable type	Variable address	Variable value
a	int	FFC0	5
b	int	FFC4	6
c	address to int	FFD4	????
r	float	FFC8	10.6
s	float	FFCC	22.3
t	address to float	FFD8	????

"Address of" operator: &

- To evaluate the addresses of variables
- It has been used before
  - **scanf()**

```
int a=5;
int *c;

c = &a;
```

Variable name	Variable type	Variable address	Variable value
a	int	FFC0	5
b	int	FFC4	6
c	address to int	FFD4	FFC0
r	float	FFC8	10.6
s	Float	FFCC	22.3
t	address to float	FFD8	????

Indirection operator: *

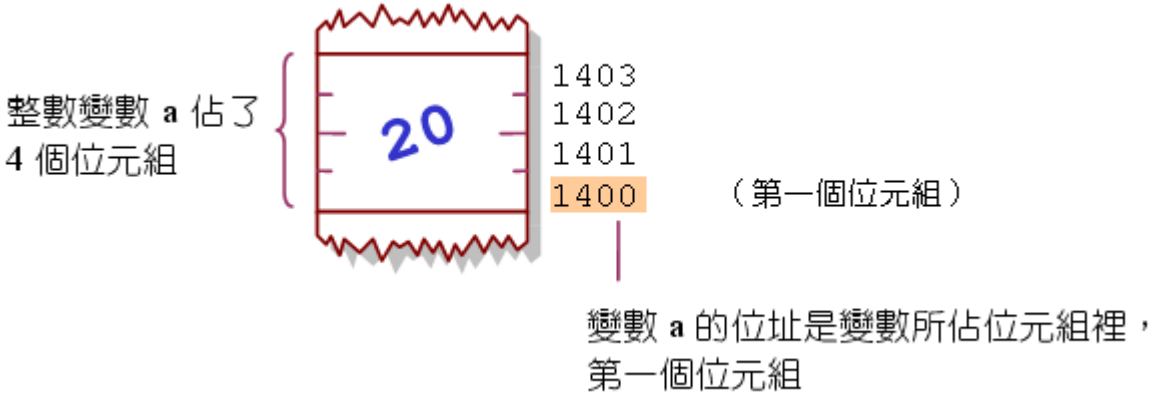
- The unary operator indicating to go the address, indicating an address to be stored in a variable's memory cell

```
int a=5;
int *c;

c = &a;
*c=100;
```

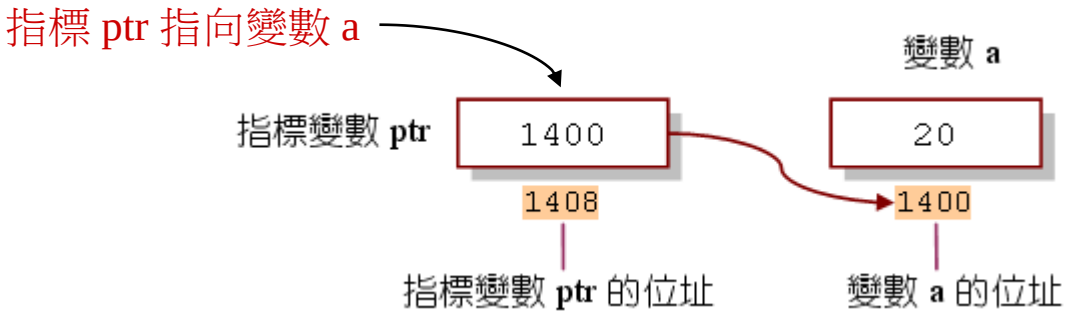
Variable name	Variable type	Variable address	Variable value
a	int	FFC0	100
b	int	FFC4	6
c	address to int	FFD4	FFC0
r	float	FFC8	10.6
s	float	FFCC	22.3
t	address to float	FFD8	????

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



指標變數的宣告

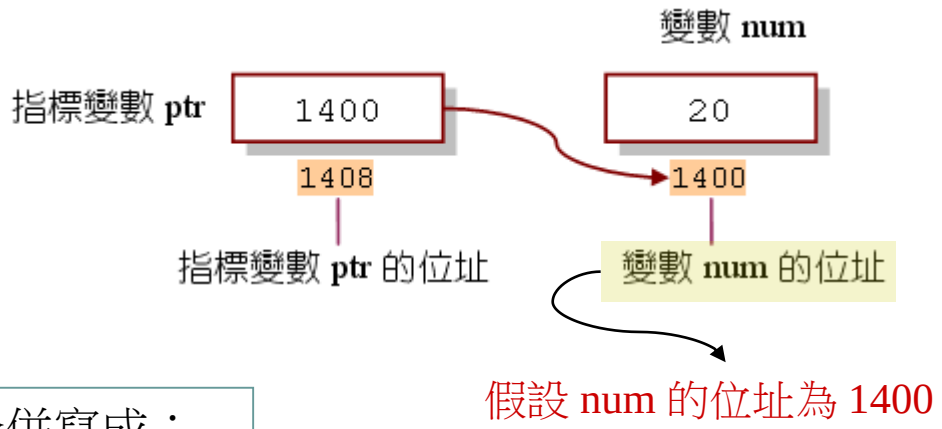
指標變數的宣告

資料型態 *指標變數;

或

資料型態* 指標變數;

- 指標變數使用的範例：
  - int num=20;
  - int *ptr;
  - ptr=#



第二行與第三行敘述也可以合併寫成：

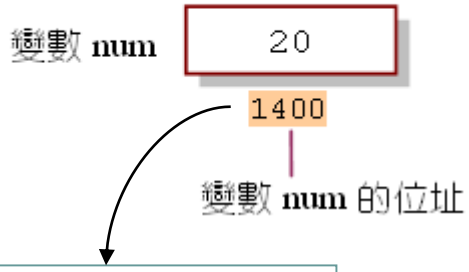
int *ptr=#

指標的使用

- 指標常用的運算有兩種：
  - 取出變數的位址，然後存放在指標裡
  - 取出指標變數所指向之變數的內容

位址運算子「&」

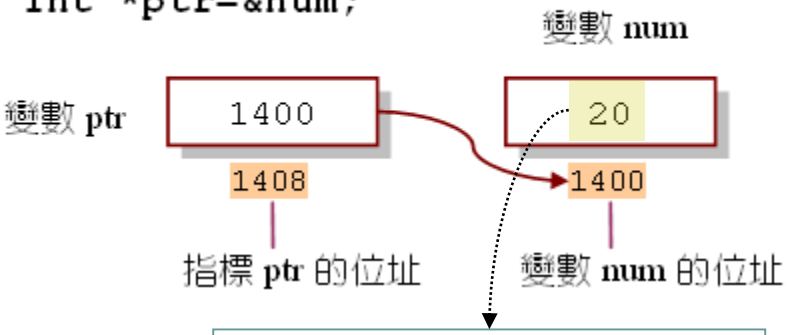
```
int num=20;
```



&num 可取出變數num的位址，即 1400

依址取值運算子「*」

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

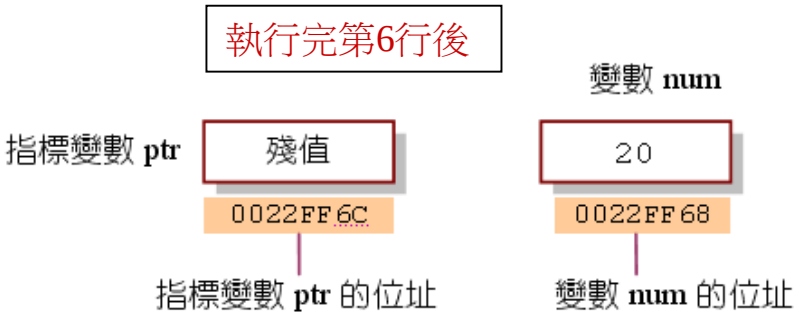


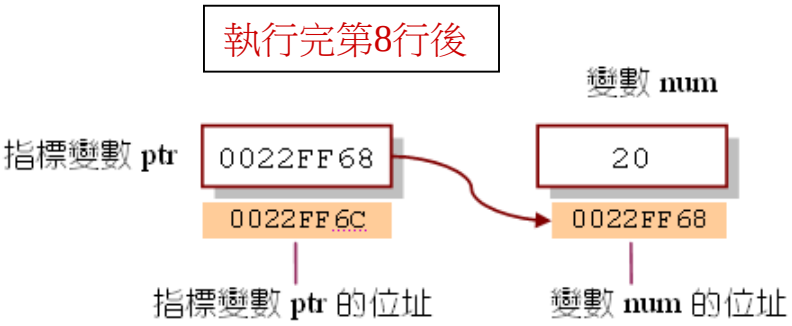
*ptr 可取出 ptr 所指向之變數 num 的內容，即 20

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



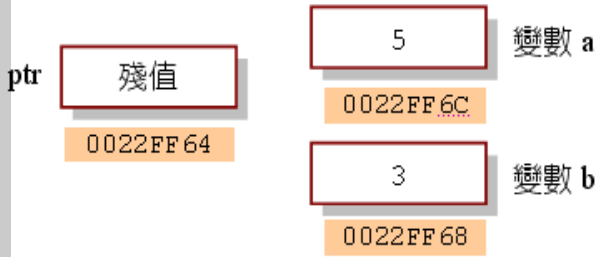


指標變數的使用範例 (2/2)

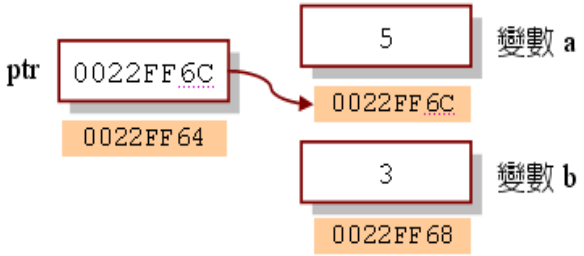
```
01  /* prog10_3, 指標變數的使用 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     int a=5,b=3;
07     int *ptr;
08
09     ptr=&a;              /* 將 a 的位址設給指標 ptr 存放 */
10     printf("&a=%p, &ptr=%p, ptr=%p, *ptr=%d\n",&a,&ptr,ptr,*ptr);
11     ptr=&b;              /* 將 b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

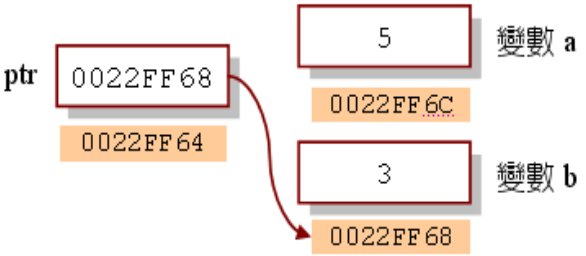
執行完第 7 行



執行完第 9 行



執行完第 11 行



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

Pointer: point to an exist object

```
int a=5;
int *c;

c = &a;
*c=100;
```

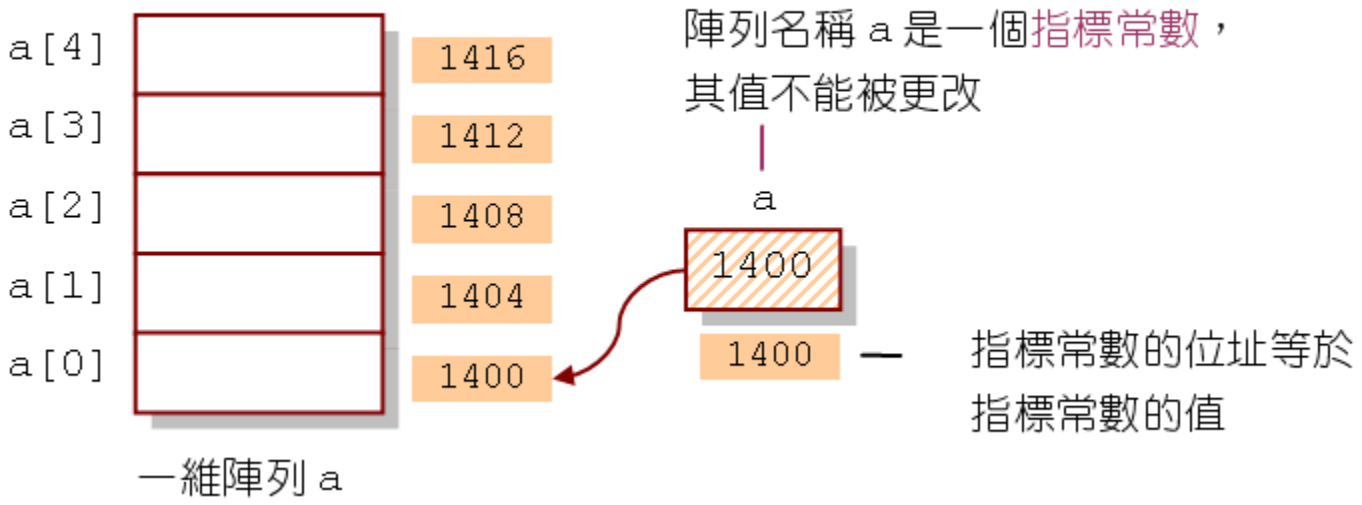
If the Pointer does not point to an exist object, what happened?

```
int *c;

*c=100;
```

指標與一維陣列

- 陣列的名稱是一個指標常數，它指向該陣列的位址

```
int a[5];
```



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：

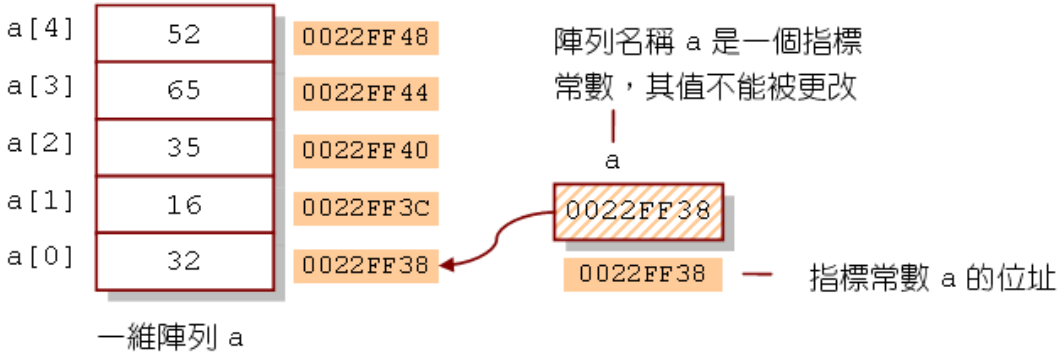
```
/* prog10_13 OUTPUT----------
a=0022FF38
&a=0022FF38
&a[0]=0022FF38
&a[1]=0022FF3C
&a[2]=0022FF40
&a[3]=0022FF44
&a[4]=0022FF48
----------------------------*/
```

指標常數 a 的值

指標常數 a 的位址

陣列元素的位址

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



How to declare character variables?

- **Data type:** `char`
- **Syntax**

  `char` *variable1, variable2, variable3, ... ;*

- **Example：**

```
...
char c1, c2, c3, c4, c5, c6, c7, c8, c9;
...
```

To assign character variables

```
c1 = 'g';
```

Assign character ‘g’ to the character variable c1

指定字元變數c1的內容為字元 ‘g’

Notice: Use single quote :「 ‘」, Not「 “」

注意：要用單引號「 ‘」，而非雙引號「 “」

Escape Sequences

- Such as \n, \r
  - These are regarded as a single character in C.
  - So the following assignment is legal:

```
c3 = '\n';
```

視為單一字元(single character) ，反斜線(\)後不可有空白字元。

putchar ( )

- 將傳入參數(字元)輸出，印至標準輸出單元(螢幕)
- 語法(Syntax)

```
putchar (character);
```

- 範例：

```
putchar(c2);
putchar('y');
```

? putchar(32)

- 字元變數(char)其實在C裡頭視為整數變數(int) 。

```
putchar(32);
```

ASCII 值 32對應為空白字元(space) (在以ASCII編碼的系統時，才有此對應關係)

```
putchar(' ');
```

這樣比較好

getchar ( )

- 自標準輸入(鍵盤)讀入一個字元。
- 語法(Syntax)

```
getchar();
```

- 範例

```
c6 = getchar();
```

Keyboard

Assignment

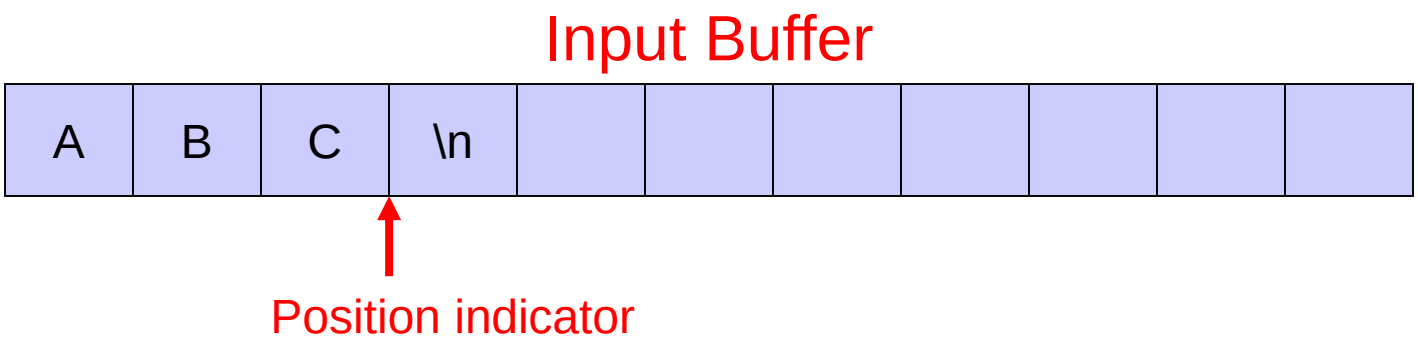
Input from the keyboard

- 並非所有的鍵盤輸入訊息都會立刻被執行，為增進系統效率，有些輸入訊息會先被儲存至輸入緩衝區(**input buffer**) ，等候其他系統函式處理。

- ***Input Buffer (輸入緩衝區)***
  - **A portion of memory reserved for temporarily holding information that is being transferred.**
- **Position indicator**
  - **keeps track of the point at which no further information has been read.**

Input Buffer and `getchar()`

- The `getchar` function works with the buffer position indicator to retrieve the next character in the buffer and advance the position indicator



fflush ( )

- **To flush or empty the buffer after obtaining the character(s) of interesting.**
- **Syntax**

```
fflush(stdin);
```

What is stdin?

- `stdin` is defined to point the standard input stream.
  - 指向標準輸入(鍵盤)
- stdin is a keyword defined in stdio.h. Do not use it as a variable name. (由於stdin已在stdio.h中定義，若拿來當作變數名(identifier) ，編譯器將會出現錯誤訊息。)

What is a string?

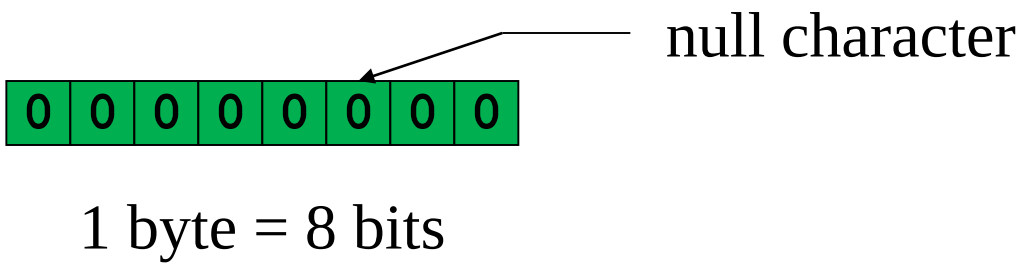
- A string is an array of character including the terminating null (`\0`) character.
- For instance,

```
bb[0]='C';
bb[1]='a';
bb[2]='t';
bb[3]='\0';
```

escape sequence → the null character

The null character

- The character, \0, is called the null character
  - It is stored in memory as 1 byte with all of the bits set to 0.



- Should not be confused with the **NULL pointer**

Double quotes " "

- When more than one character is written, we use double quotes to surround the characters.
  - C automatically adds the terminating null character when it is stored in memory.

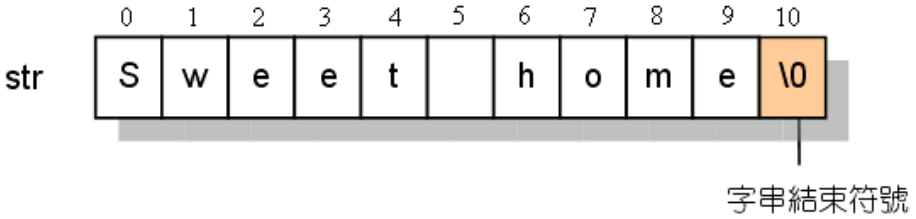
Array & String

- 字元以單引號包圍，而字串則是以雙引號包圍：
  - 'a' /* a character: a */
  - "a" /* a string: a */
  - "Sweet home" /* a string: Sweet home */
- 下面是字串宣告的語法：

字串宣告的語法

```
char 字元陣列名稱[陣列大小] = 字串常數;
char str[100]="A string";
```

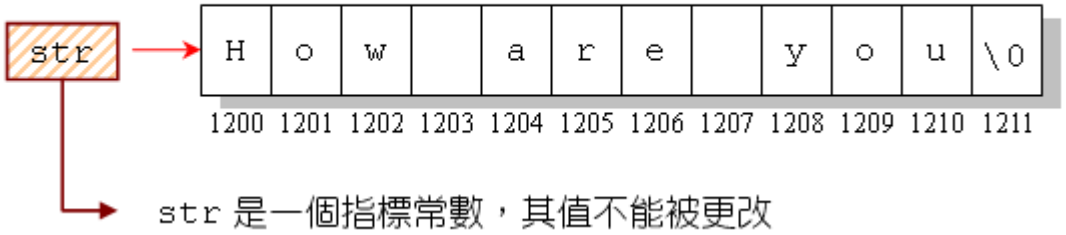
```
char str[]="Sweet home";
```



以指標變數指向字串 (1/2)

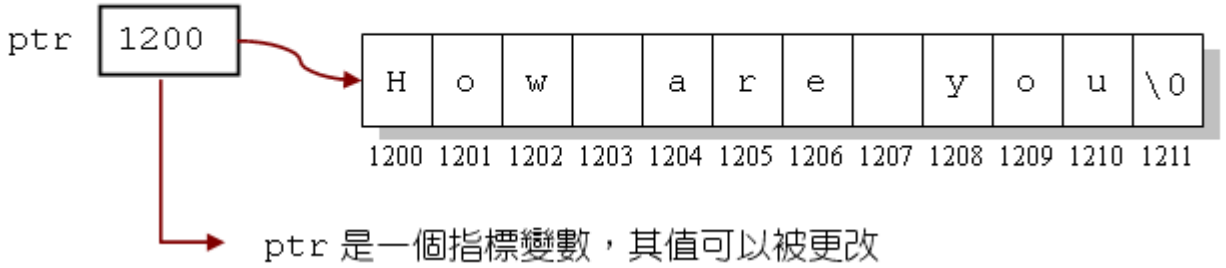
- 利用字元陣列來儲存字串：

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



- 利用指標指向字串：

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



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

Input and Output of String

- Read a string from keyboard 從鍵盤讀取字串資料。
- 函式讀取從鍵盤直到按下enter鍵之前**所鍵入的所有字元**
- 最後由鍵盤鍵入的 ‘\n’(New Line)字元將不會被讀入，並自動以空白字元(‘\0’)填入，以作為字串結束。

```
gets(address);
```

該位址表示讀入字串放置處的記憶體之開始位置。

gets() 的格式

```
gets(字元陣列名稱); //gets(a string array name);
```

```
char a[40];
gets(a);
```

keyboard

Input and Output of String

puts()函式會自動在印出字串後加上換行

puts() 的格式

```
puts(字元陣列名稱); //puts(a string array name);
```

或者是

```
puts(字串常數);
```

An Example

```
01  /* prog9_21, 輸入及印出字串 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  int main(void)
05  {
06     char name[15];     /* 宣告字元陣列 name */
07
08     puts("What's your name?");
09     gets(name);        /* 利用 gets()讀入字串，並寫入字元陣列 name 裡 */
10     puts("Hi!");
11     puts(name);        /* 印出字元陣列 name 的內容 */
12     puts("How are you?");
13     system("pause");
14     return 0;
15  }
```

An Example

```
/* prog9_22 OUTPUT----------
請輸入一個字串: Happy Birthday
轉換成大寫後: HAPPY BIRTHDAY
----------------------------*/
```

```
01  /* prog9_22, 將字串裡小寫字母轉換成大寫 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void toUpper(char s[]);   /* 宣告函數 toUpper()的原型 */
05  int main(void)
06  {
07     char str[15];          /* 宣告可容納 15 個字元的陣列 str */
08
09     printf("請輸入一個字串: ");
10     gets(str);             /* 輸入字串 */
11     toUpper(str);          /* 呼叫 toUpper() 函數 */
12     printf("轉換成大寫後: %s\n",str);      /* 印出 str 字串的內容 */
13
14     system("pause");
15     return 0;
16  }
17
```

An Example : toUpper()

```
18  void toUpper(char s[])
19  {
20     int i=0;
21     while(s[i]!='\0')        /* 如果 s[i] 不等於\0，則執行下面的敘述 */
22     {
23        if(s[i]>=97 && s[i]<=122) /* 如果是小寫字母 */
24           s[i]=s[i]-32;      /* 把小寫字母的 ASCII 碼減 32，變成大寫 */
25        i++;
26     }
27  }
```

```
/* prog9_22 OUTPUT----------

請輸入一個字串: Happy Birthday
轉換成大寫後: HAPPY BIRTHDAY
----------------------------*/
```

```c
/* Program for Lesson  7_1 */
#include <stdio.h>
#include <string.h>
void main(void)
{
  char aa, bb[4], cc[100], dd[100];
  FILE *outfile;
  outfile=fopen("L7_1.out","w");

  printf ("**********  Section 1 - Initializing ************\n");
  aa='g';
  bb[0]='C';
  bb[1]='a';
  bb[2]='t';
  bb[3]='\0';

  strcpy(cc,"This is a string constant, also called a string literal.");
  strcpy(dd,cc);

...
```

Single character

Character array → String

Single quotes for singe character

The string of characters enclosed in double quotes is treated as an address in C.

Initialization for a string

- Character by character

```
bb[0]='C';
bb[1]='a';
bb[2]='t';
bb[3]='\0';
```

- C library function
  - **strcpy( )**

A common error!!

```
char aa, bb[4], cc[100], dd[100];
...
cc="This is a string constant, also called....";
```

C sees an address for the string literal given on the right side of the assignment statement.

Such a statement causes C to try to store an address in the location indicated by cc.

The content of cc is fixed after it has been declared → could not be changed

```
char aa, ee[2];
aa='g';
strcpy(ee, "g");
```

Memory for character variable aa	g	
Memory for character variable ee[ ]	g	\0

The different between string and character variables

Array vs. Pointer

```
char aa, bb[4], cc[100], dd[100];
```

where the contents of variables, **bb**, **cc**, and **dd** are addresses, but they are not pointer variables.

```
dd = cc;
```

```
dd = "A sample string";
```

`strlen( )` function

- The function **`strlen`** is used to determine the number of characters actually stored in a particular character array.
  - Not the size of the character array.
- Syntax:

strlen(*address*);

Return the number of bytes or memory cells up to but not including the null character in the string

`%s` conversion specification

- With **`printf()`**
  - **`%s`** indicates to **print until a null character** is encountered.
  - To print an *entire string*
- With **`scanf()`**
  - **`%s`** means to read until **a white-space character** is encountered.
  - To read only ***one word (只有讀入一個word)***

9.3指標與函數

Functions – Call by address

I-Fen Chao

```
int a=5;
int *c;

c = &a;
*c=100;
```

```
int a=5;
int *c=&a;

*c=100;
```

(全域範圍)

```
#include <stdio.h>
#include <stdlib.h>

void main(void)
{
    int a=5;
    int *c;

    c=&a;
    *c=100;

    system("pause");

}
```

100 %

監看式 1

名稱	值
a	-858993460
&a	0x002efc58
c	0xcccccccc
&c	0x002efc4c

```
#include <stdio.h>
#include <stdlib.h>

void main(void)
{
    int a=5;
    int *c;

    c=&a;
    *c=100;

    system("pause");

}
```

100 %

監看式 1

名稱	值
a	5
&a	0x002efc58
c	0x002efc58
&c	0x002efc4c

```
#include <stdio.h>
#include <stdlib.h>

void main(void)
{
    int a=5;
    int *c;

    c=&a;
    *c=100;

    system("pause");

}
```

100 %

監看式 1

名稱	值
a	100
&a	0x002efc58
c	0x002efc58
&c	0x002efc4c

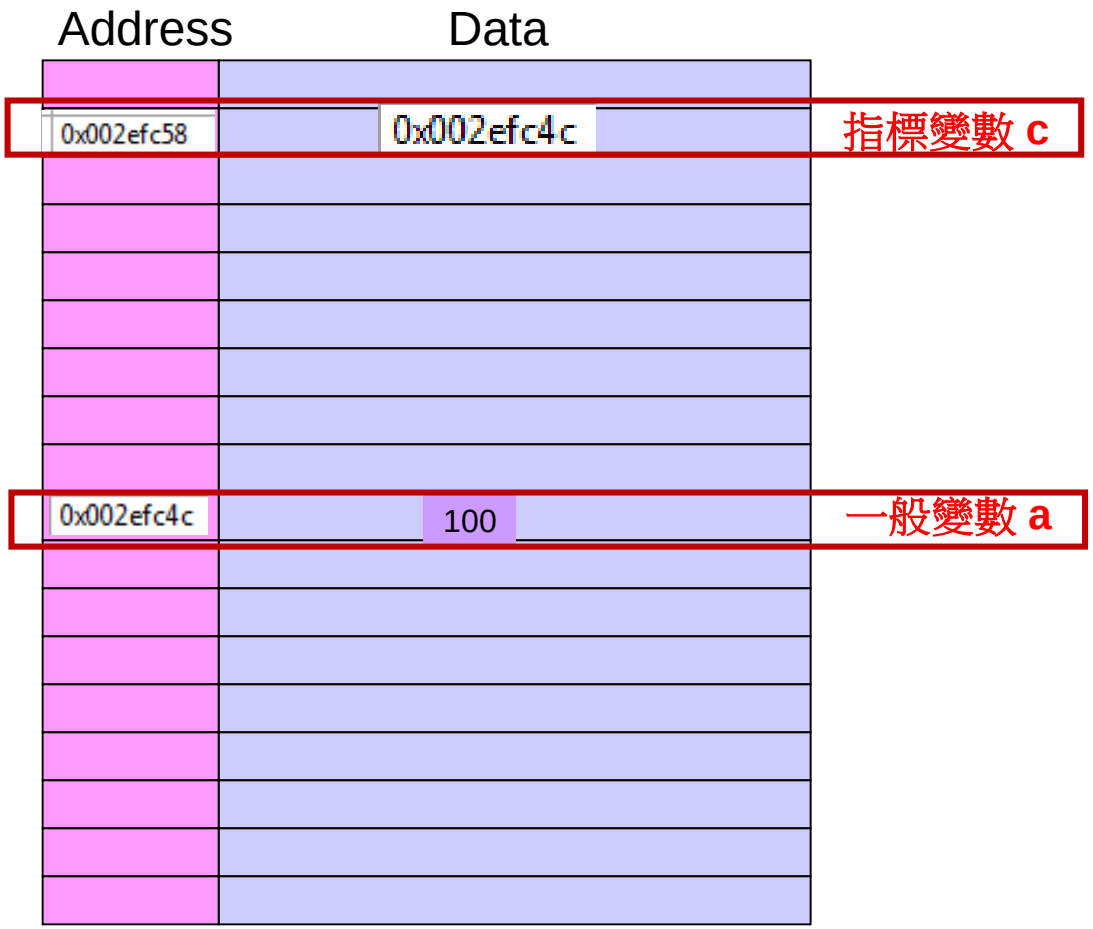
```
int a=5;
int *c;

c = &a;
*c=100;
```

```
#include <stdio.h>
#include <stdlib.h>

void main(void)
{
    int a=5;
    int *c;

    c=&a;
    *c=100;

    system("pause");

}
```

100 %

監看式 1

名稱	值
a	5
&a	0x002efc58
c	0x002efc58
&c	0x002efc4c



指標變數

- 宣告:

```
e.g.  int *point_to_int;
      float *point_to_float;
```

- 初始化，給值(assign address):

```
int x=5;
point_to_int = &x;  // 指標變數儲存x的地址
int *p1 = &x; //指標變數初始化為x的地址
```

- 藉由指標變數，修改其儲存的地址的內容

```
*point_to_int  = 100;  // x的值從5改變為100
```

- What are the values?

```
point_to_int, *point_to_int, &point_to_int
```

Example: the address of a variable

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   int *ptr,num=20;

   ptr=&num;
   printf("num=%d, &num=%p\n",num,&num);
   printf("*ptr=%d, ptr=%p, &ptr=%p\n",*ptr,ptr,&ptr);
   system("pause");
   return 0;
}
```

```
num=20, &num=0038F978
*ptr=20, ptr=0038F978, &ptr=0038F984
請按任意鍵繼續 . . .
```

Example: the address of a variable

```
/* Print the address of a variable*/
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
   int a,b=5;          /* variable a without initialization*/
   double c=3.14;

   printf("a=%4d, sizeof(a)=%d, address is=0x%p\n",a, sizeof(a),&a);
   printf("b=%4d, sizeof(b)=%d, address is=0x%d\n",b, sizeof(b),&b);
   printf("c=%4.2f, sizeof(c)=%d, address is=0x%d\n",c, sizeof(c),&c);
   system("pause");
}
```

```
a=-858993460, sizeof(a)=4, address is=0x003BFD28
b=   5, sizeof(b)=4, address is=0x3931420
c=3.14, sizeof(c)=8, address is=0x3931404
請按任意鍵繼續 . . .
```

```
a=-858993460, sizeof(a)=4, address is=0x003EFBA8
b=   5, sizeof(b)=4, address is=0x4127644
c=3.14, sizeof(c)=8, address is=0x4127628
請按任意鍵繼續 . . .
```

Example: the address of a variable

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

```
&a=002DFB44, &ptr=002DFB2C, ptr=002DFB44, *ptr=5
&b=002DFB38, &ptr=002DFB2C, ptr=002DFB38, *ptr=3
請按任意鍵繼續 . . .
```

UltraEdit-32 - [C:\lect\chap5\15-5.c]

File Edit Search Project View Format Column Macro Advanced Window Help

15-4.c 15-5.c

```c
#include <stdio.h>
void function1(int a, int b, double r, double s, int *c, double *t);

void main(void)
{
   int      i=5,     j=6, k;
   double   x=10.6,  y=22.3, z;

   printf(" i =%d \n\r j = %d \n\r x = %lf\n\r y = %lf \n\n",
          i, j, x, y);

   function1(i,j,x,y,&k,&z);

   printf(" k = %d \n\r z = %lf \n\n", k, z);

   printf(" Address of k = %p\n\r Address of z = %p \n", &k, &z);
}

vo                int b, double r, double s, int *c, double *t)
{
```

conversion specifier **%p**

OUTPUT

```
Address of k = 0063FDD4
Address of z = 0063FDDC
```

For He... Mod: 2001/10/26 01:29:32AM File Size: 716 INS

開始 Microsoft PowerPoint - [Le... UltraEdit-32 - [C:\lect\chap... MS-DOS 模式 En AM 01:29

more about scanf( )

```
scanf("%d", &k);
```

其實傳入**scanf()**函式的是變數**k**的位置

```
scanf(..., int *c, ...)
{
   ...
   *c = xxx; //讀入的值
   ...
}
```

透過此方式即可將不只一個讀入值回傳給原函式

Passing Arguments to Functions

- There are three ways to pass arguments to a function
  - call-by-value: `void function1(int a);`
  - call-by-address: `void function2(int *a);`
    - Pass address of argument using & operator
      - int b;
      - function(&b);
    - Allows you to change actual location in memory (直接存取變數原本的記憶體內容)

UltraEdit-32 - [C:\lect\chap5\15-5.c]

File Edit Search Project View Format Column Macro Advanced Window Help

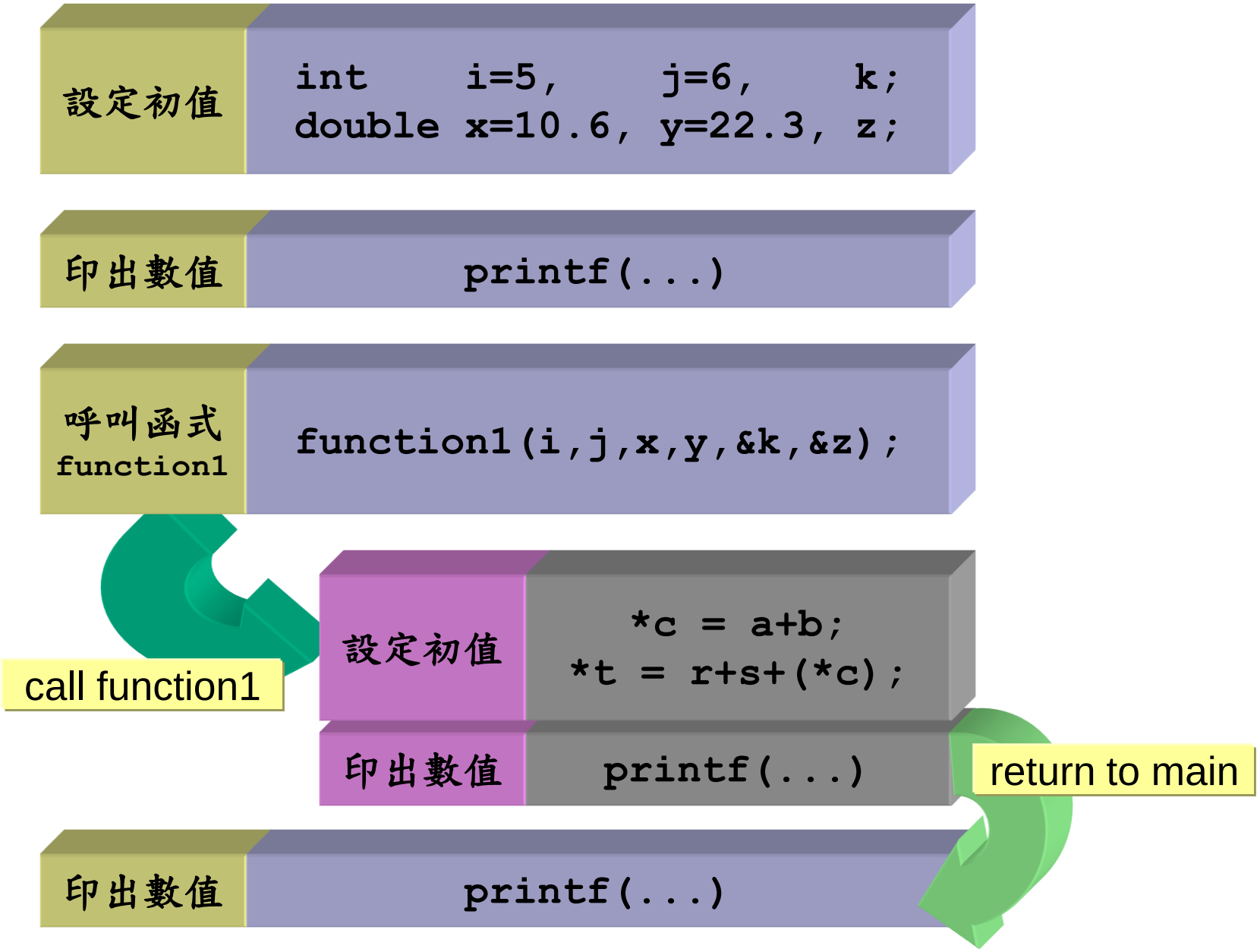
15-4.c 15-5.c

```c
#include <stdio.h>
void function1(int a, int b, double r, double s, int *c, double *t);

void main(void)
{
   int      i=5,     j=6, k;
   double   x=10.6,  y=22.3, z;

   printf(" i =%d \n\r j = %d \n
          i, j, x, y);

   function1(i,j,x,y,&k,&z);

   printf(" k = %d \n\r z = %lf \n\n", k, z);

   printf(" Address of k = %p\n\r Address of z = %p \n", &k, &z);
}

void function1(int a, int b, double r, double s, int *c, double *t)
{
   *c = a+b;
```

The addresses of k and z were passed into function1

For Help, press F1 | Ln 12, Col. 1, CW | DOS | Mod: 2001/10/26 01:29:32AM | File Size: 716 | INS

開始 | Microsoft PowerPoint - [Le... | UltraEdit-32 - [C:\lect\chap... | MS-DOS 模式 | En | AM 02:11


設定初值
int i=5, j=6, k;
double x=10.6, y=22.3, z;
印出數值
printf(...)
呼叫函式
function1
function1(i,j,x,y,&k,&z);
call function1
設定初值
*c = a+b;
*t = r+s+(*c);
印出數值
printf(...)
return to main
印出數值
printf(...)

OUTPUT

```
 i = 5
 j = 6
 x = 10.600000
 y = 22.300000

*c = 11
*t = 43.900000

 k = 11
 z = 43.900000
```

values of variables in function1

values of variables in main

由main()傳至function1()的資料
The values have been transferred from main to function1

```
main()
{
    int i,j,k;
    double x,y,z;
    function1(i,j,x,y,&k,&z);

   ...
}
```

```
void function1(a,b,r,s,*c,*t)
{
    *c=a+b;
    *t=r+s+(*c);
    ...
}
```

由function1()傳回至main() 的資料
The values have been returned from function1 to main

```
main()
{
    int i,j,k;
    double x,y,z;
    function1(i,j,x,y,&k,&z);

   ...
}
```

```
void function1(a,b,r,s,*c,*t)
{
    *c=a+b;
    *t=r+s+(*c);
    ...
}
```

Call by Value vs. Call by Address

- C only copies the values of the variables in the parameter list into the function's memory region.

```
main()
{
    int i,j,k;
    double x,y,z;
    function1(i,j,x,y,&k,&z);

   ...
}
```

CALL BY VALUE

只能單向傳遞

The values are not changed!

```
void function1(a,b,r,s,*c,*t)
{
    *c=a+b;
    *t=r+s+(*c);
    ...
}
```

Call by Value vs. Call by Address

- **The function can return more than one value via Call by addrss**

```
main()
{
    int i,j,k;
    double x,y,z;
    function1(i,j,x,y,&k,&z);

   ...
}
```

CALL BY address

Values are modified!

```
void function1(a,b,r,s,*c,*t)
{
    *c=a+b;
    *t=r+s+(*c);
    ...
}
```

```
int k;
double *t;
```

```
&k = FDD2;
```

→illegal, **&k** is a value, could not present in the left side of the assignment statement

```
t = FFC4;
```

→legal, but may result in some problems

Something to remind

- Up to now, there are two ways to "return" the values to the calling function
  - use `return` statement
  - use the addresses in the parameter list

傳遞指標到函數 (1/3)

- 接收指標的函數：

接收指標之函數的語法

```
傳回值型態  函數名稱(資料型態 *指標變數)
{
    /*  函數的本體  */
}
```

接收的是變數的位址

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

傳遞變數的位址

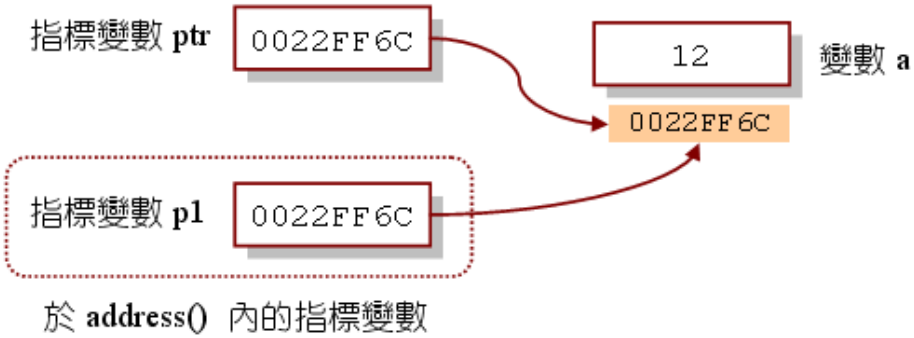
```
void func(int *ptr)
{
    /*  函數的本體  */
}
```

接收變數的位址

傳遞指標到函數 (2/3)

- 傳遞指標到函數的範例 ：

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



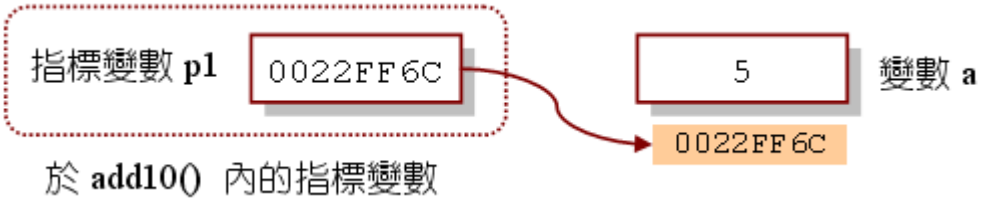

```
01  /* prog10_7, 傳遞指標到函數裡 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void address(int *);
05  int main(void)
06  {
07     int a=12;
08     int *ptr=&a;
09
10     address(&a);            /* 將 a 的位址傳入 address() 函數中 */
11     address(ptr);           /* 將 ptr 傳入 address() 函數中 */
12
13     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)

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

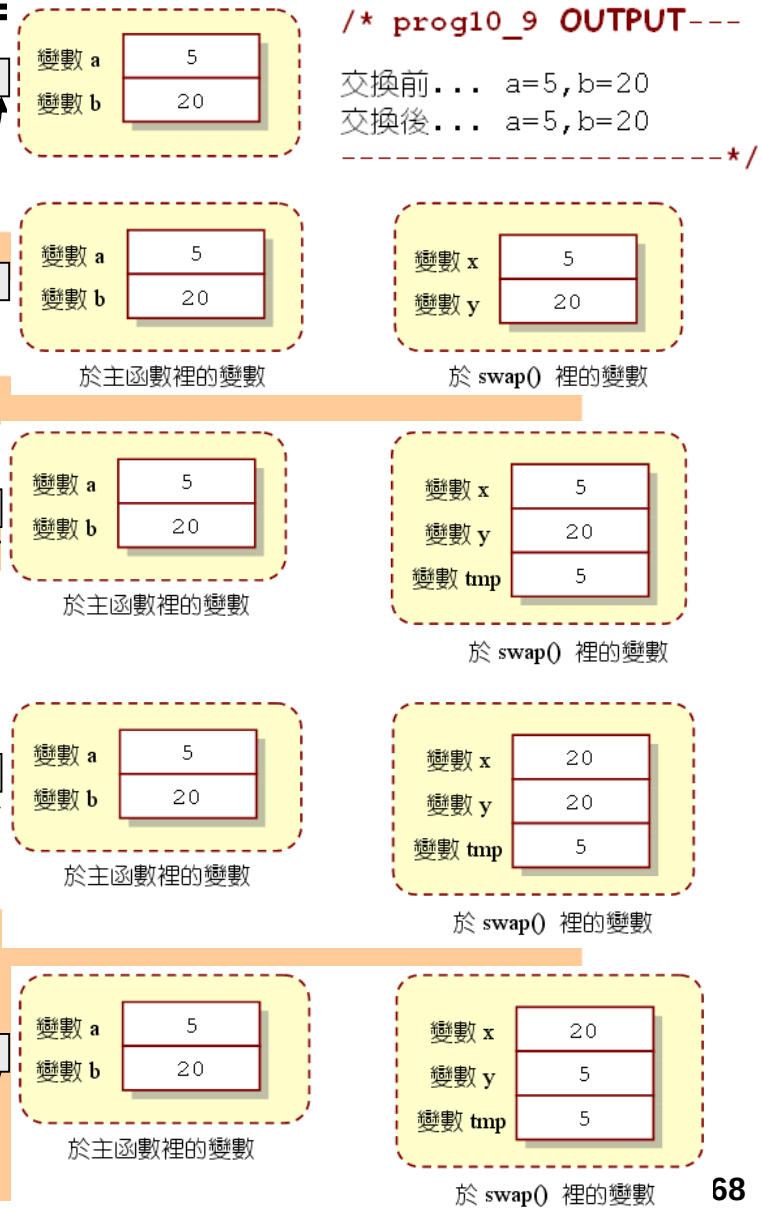
- 傳遞指標的應用：

```
/* prog10_8, 傳遞指標的應用 */
#include <stdio.h>
#include <stdlib.h>
void add10(int *);
int main(void)
{
   int a=5;
   printf("呼叫 add10()之前,a=%d\n",a);
   add10(&a);                /* 呼叫 add10()函數 */
   printf("呼叫 add10()之後,a=%d\n",a);
   system("pause");
   return 0;
}
void add10(int *p1)
{
   *p1=*p1+10;
}
```



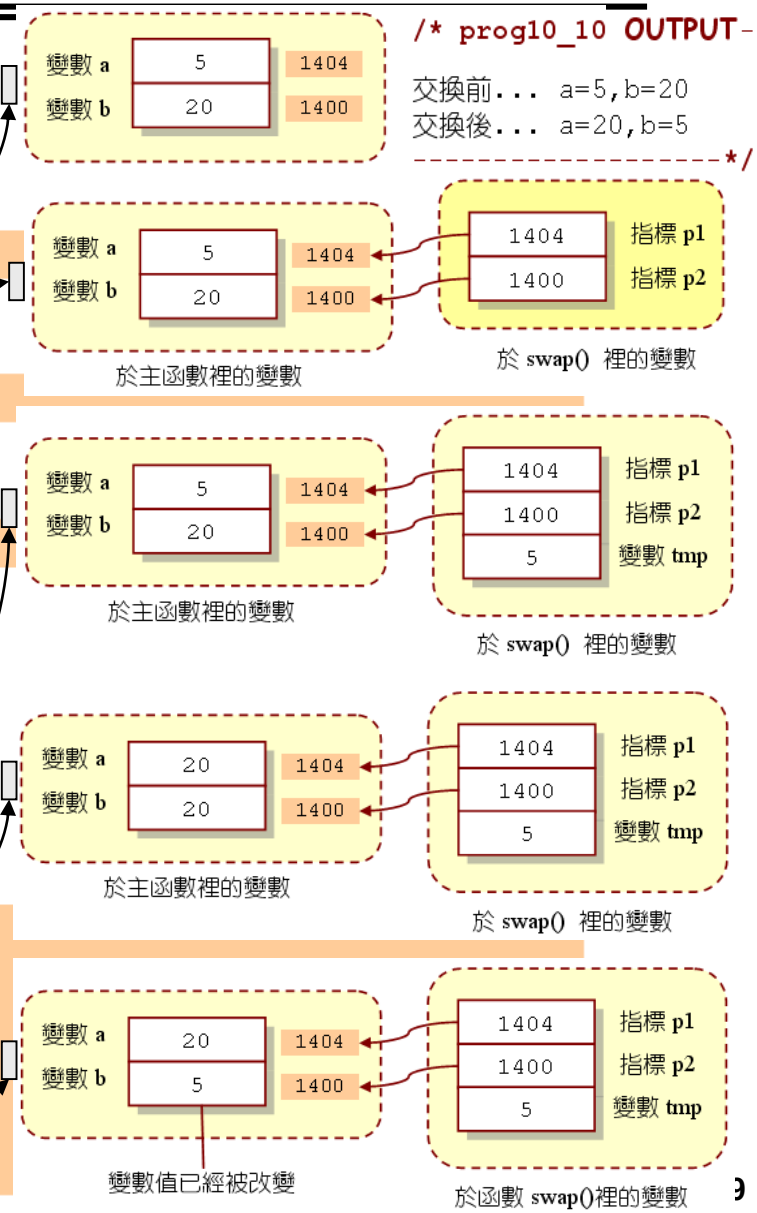
變數值的互換 (錯誤)

```
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  }
```



變數值的互換 (正確)

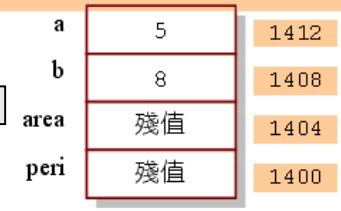
```
01  /* prog10_10, 將 a 與 b 值互換(正確範例) */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void swap(int *,int *);
05  int main(void)
06  {
07     int a=5,b=20;
08     printf("交換前... ");
09     printf("a=%d,b=%d\n",a,b);
10     swap(&a,&b);
11     printf("交換後... ");
12     printf("a=%d,b=%d\n",a,b);
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  }
```

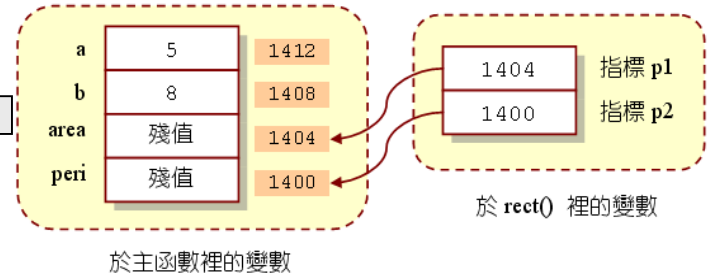


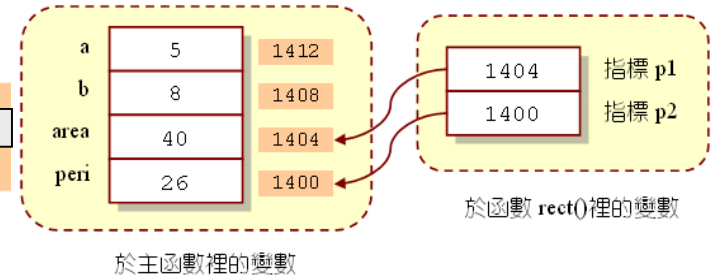
傳回多個數值的函數

```
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",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  }
```

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







Practice – call by value

- write function increase_add()
  - with three integer parameter: a, b, c
  - a++; b++; c++;
  - 印出: a, b, c的位址及值
  - return a+b+c

1. **Call by value**
2. **Call by address**
3. **In main program: call function by:**

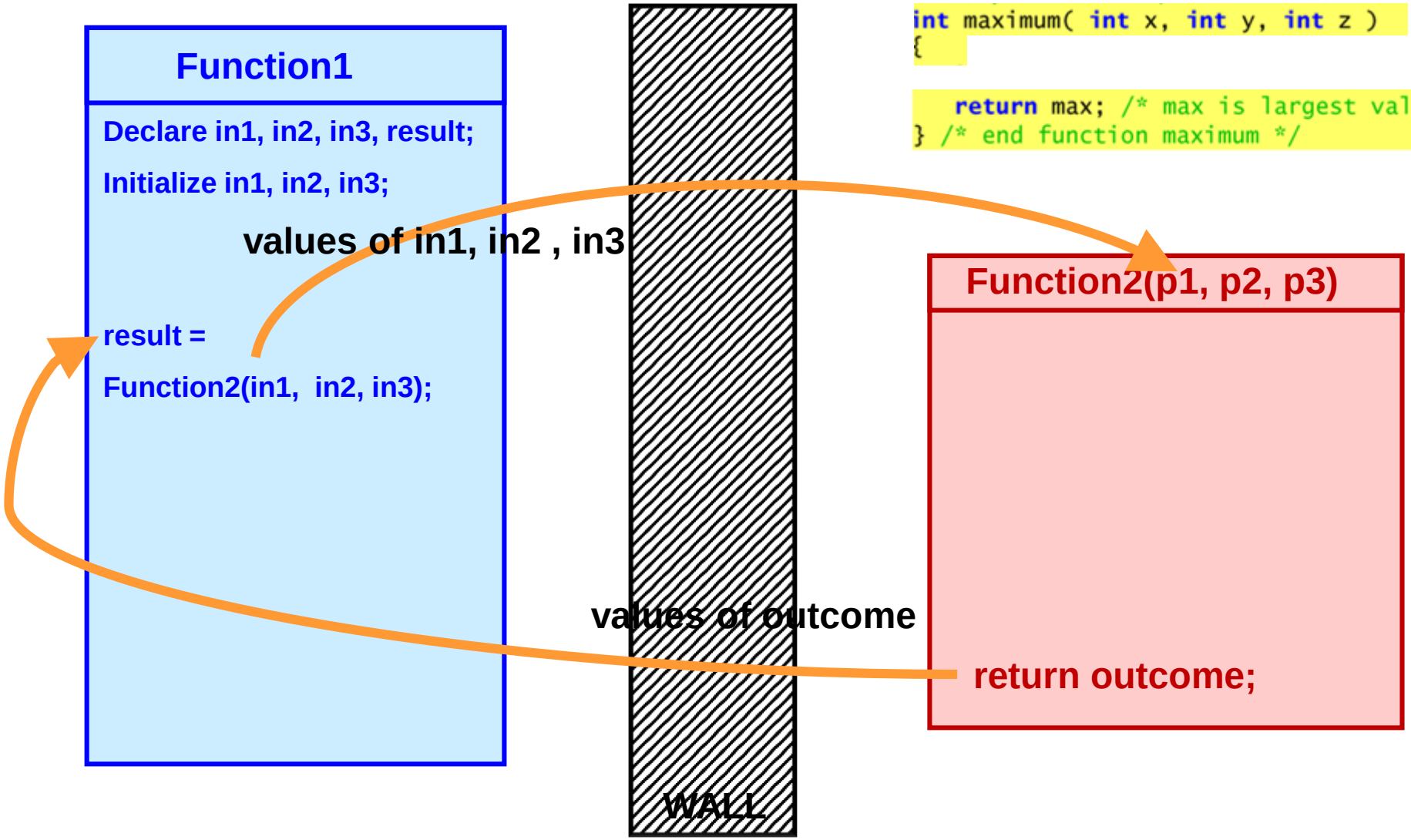
increase_add(x, y, z);

前後印出**x, y, z** 的位址及值

Function1 call Function2 – call by value

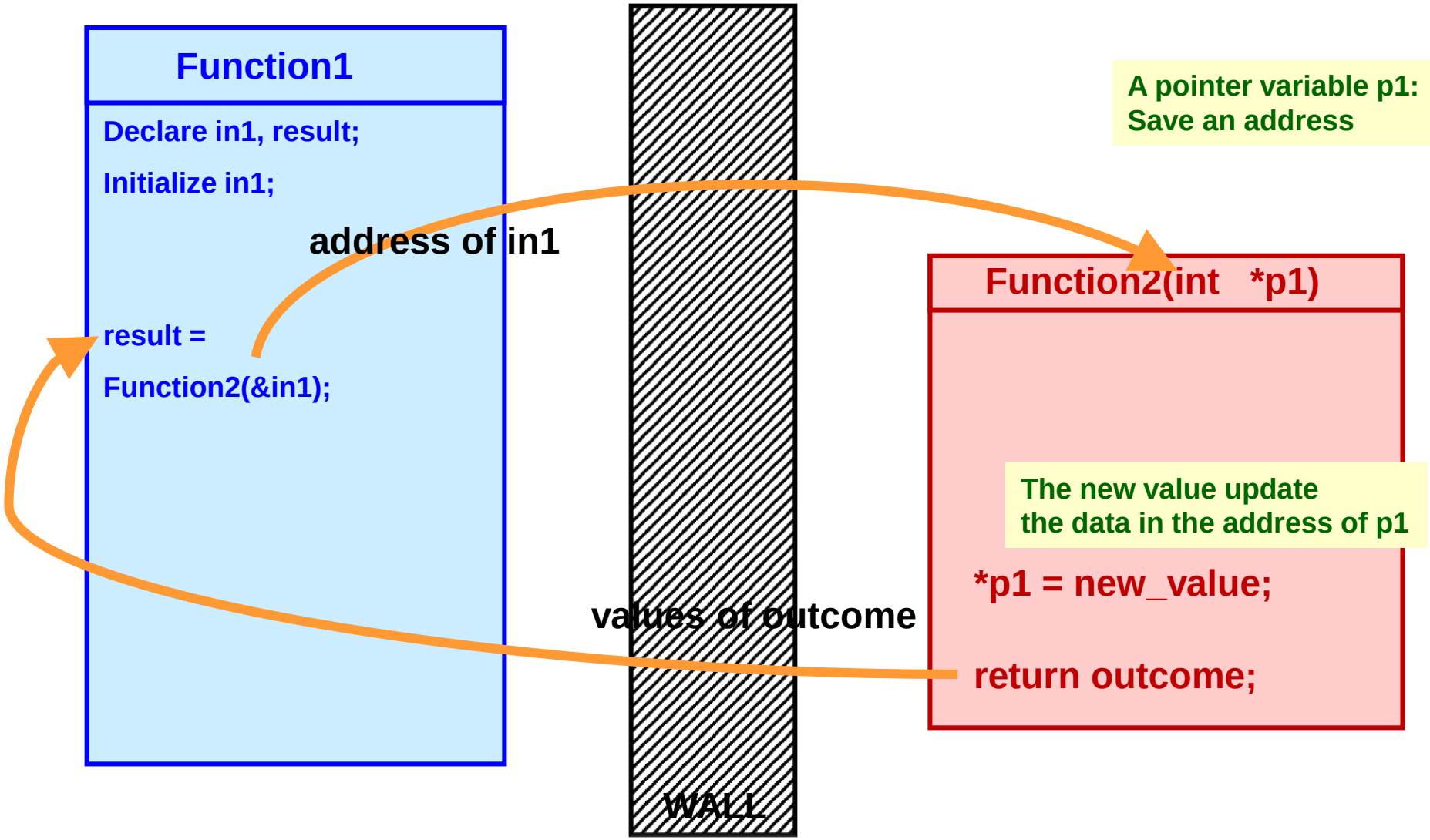
```
int maximum( int x, int y, int z ); /* function prototype */
```

```
int maximum( int x, int y, int z )
{

   return max; /* max is largest value
} /* end function maximum */
```



Independent of two worlds (Modules)!!

Function1 call Function2 – call by address



Independent of two worlds (Modules)!!

NOTE: Call by address

- **Step 1:** 宣告:參數宣告成指標變數，因為要存地址

```
return_data_type fun_name( int *p1, float *p2 );
```

- **Step 2:** call function in main() or other functions

```
int x; float y;
fun_name( &x, &y );
```

呼叫此function時，傳地址過去，告訴它，變數的位址，該function才能改變其值

- **Step3:** Function 程式撰寫:

```
return_data_type fun_name(int *p1, float *p2)
{    ....
     *p1= new value;
     *p2= new value;
}
```

利用 *variable，來修改傳入的指標參數的該地址的內容

Pass an array: Pass by address

```
01  /* prog9_12, 傳遞一維陣列到函數裡 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define SIZE 4
05  void show(int arr[]);          /* 宣告函數 show()的原型 */
06  int main(void)
07  {
08     int A[SIZE]={5,3,6,1};      /* 設定陣列 A 的初值 */
09     printf("陣列的內容為: ");
10     show(A);                    /* 呼叫函數 show() */
11     system("pause");
12     return 0;
13  }
14  void show(int arr[])           /* 函數 show()的定義 */
15  {
16     int i;
17     for(i=0;i<SIZE;i++)
18        printf("%d ",arr[i]);    /* 印出陣列內容 */
19     printf("\n");
20  }
```

傳入的是陣列的**地址**

Pass an string array: Pass by address

傳入的是字串陣列的**地址**

```
01  /* prog9_22, 將字串裡小寫字母轉換成大寫 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  void toUpper(char s[]);   /* 宣告函數 toUpper()的原型 */
05  int main(void)
06  {
07     char str[15];          /* 宣告可容納 15 個字元的陣列 str */
08
09     printf("請輸入一個字串: ");
10     gets(str);             /* 輸入字串 */
11     toUpper(str);          /* 呼叫 toUpper() 函數 */
12     printf("轉換成大寫後: %s\n",str);      /* 印出 str 字串的內容 */
13
14     system("pause");
15     return 0;
16  }
17
18  void toUpper(char s[])
19  {
20     int i=0;
21     while(s[i]!='\0')       /* 如果 s[i] 不等於\0，則執行下面的敘述 */
22     {
23        if(s[i]>=97 && s[i]<=122) /* 如果是小寫字母 */
24           s[i]=s[i]-32;     /* 把小寫字母的 ASCII 碼減 32，變成大寫 */
25        i++;
26     }
27  }
```