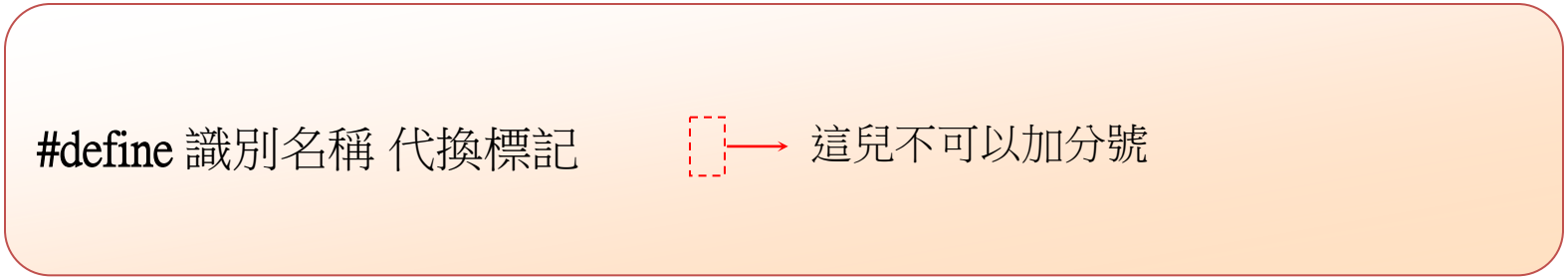
C 程式語言 7

巨集替換指令

- #define可用來定義巨集，也就是以一個識別字，取代一連串動作或程式敘述



巨集替換指令

- 巨集替換指令（Macro Substitution）#define可以定義新關鍵字或新增參數來建立巨集函數：

- 上述語法是將後面的文字內容替換成前面的名稱，替換內容可以是常數值，如下所示：

```
#define SIZE        8
#define MSG         “歡迎光臨”
```

巨集替換指令 – 替換運算式

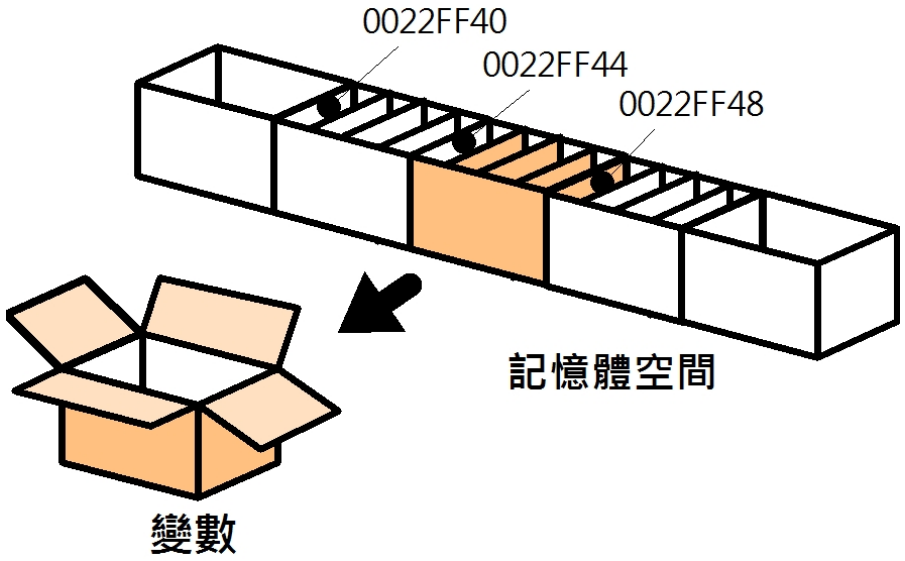
- 替換內容也可以是C語言的運算式或程式敘述，如下所示：

```
#define ONE      1
#define TWO      ONE + ONE
```

- 上述巨集指令定義ONE、TWO，ONE的值是1，在定義巨集ONE後，就可以馬上使用ONE來定義TWO。

電腦的記憶體位址

- 記憶體空間如同一排大樓信箱，每一個儲存單位擁有數字編號的「位址」（Address），即信箱上的門牌號碼，在每一單位的記憶體內容是資料，儲存資料佔用的記憶體空間大小，需視使用的資料型態而定：

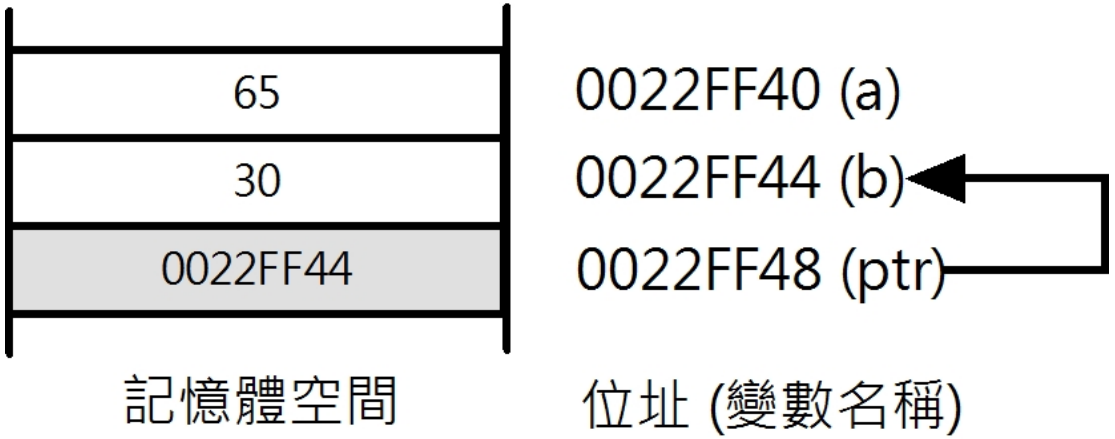


C語言的指標

- 「指標」（Pointers）是 C 語言的低階程式處理功能，可以直接存取電腦的記憶體位址。指標是一種變數，只是變數內容不是常數值，而是其他變數的「位址」，其值是指向其他變數的位址，能夠讓我們間接取得其他變數的值。

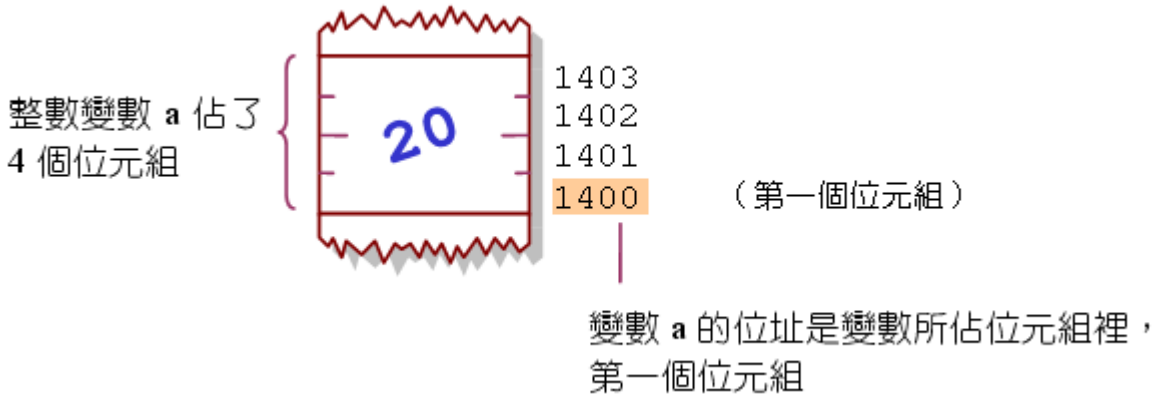
C語言的指標

- 3個變數a、b和ptr，變數值分別為65、30和0022FF44，ptr變數值0022FF44是變數b的記憶體位址，ptr是一個指標，為什麼叫指標？因為它是一個指向其他變數位址的變數，可以引導我們找到其他變數的值，以此例就是變數b的值：

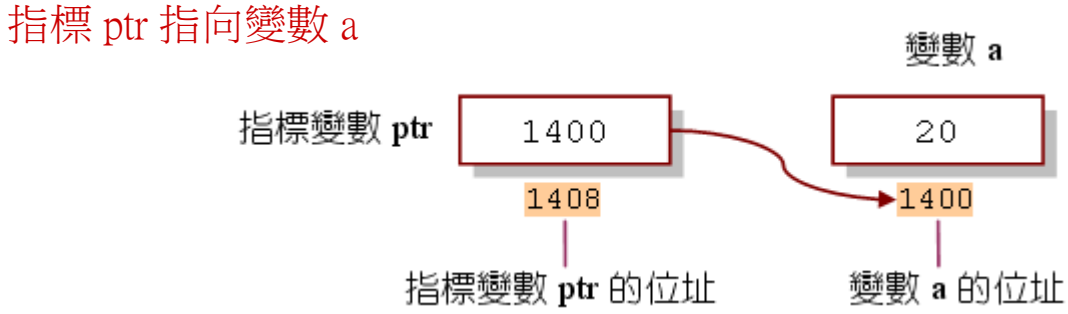


變數的位址

- 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

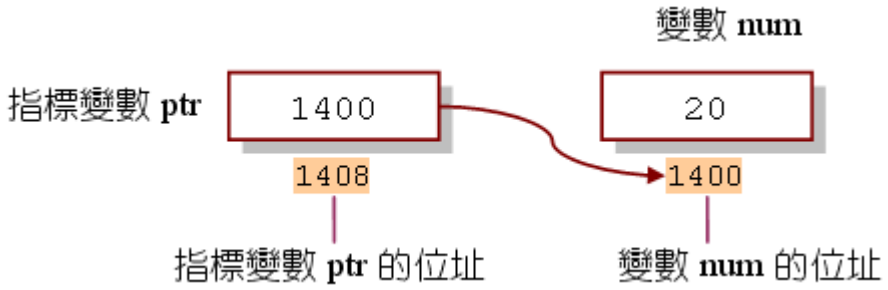
資料型態 *指標變數;

或

資料型態* 指標變數;

– 指標變數使用的範例：

- int num=20;
- int *ptr;
- ptr=#



假設 num 的位址為 1400

指標變數的使用範例

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int *ptr, num=20;
  ptr=&num;                                  /* 將num的位址設給指標ptr存放 */
  printf("num=%d, &num=%x\n",num,&num);
  printf("*ptr=%d, ptr=%x, &ptr=%x\n",*ptr,ptr,&ptr);

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

指標變數所佔的位元組

- 查詢指標變數所佔的位元組：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int *ptri;                    /* 宣告指向整數的指標ptri */
  char *ptrc;                   /* 宣告指向字元的指標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));

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

指標操作的練習

```
#include <stdio.h>
#include <stdlib.h>
int main(void){
  int a=5,b=10;
  int *ptr1,*ptr2;
  ptr1=&a;                                   /* 將ptr1設為a的位址 */
  ptr2=&b;                                   /* 將ptr2設為b的位址 */
  printf("a=%p, b=%p, \nptr1=%p, ptr2=%p\n",&a,&b,ptr1,ptr2);
  *ptr1=7;
  *ptr2=32;
  a=17;
  ptr1=ptr2;
  *ptr1=9;
  ptr1=&a;
  a=64;
  *ptr2=*ptr1+5;
  ptr2=&a;
  printf("a=%2d, b=%2d, *ptr1=%2d, *ptr2=%2d\n",a,b,*ptr1,*ptr2);
  printf("ptr1=%p, ptr2=%p\n",ptr1,ptr2);
  system("pause");
  return 0;
}
```

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：

```
#include <stdio.h>
#include <stdlib.h>
int main(void) {
  int a1=100, *ptri;
  float a2=3.2f, *ptrf;
  ptri=&a2;          /* 錯誤，將int型態的指標指向float型態的變數 */
  ptrf=&a1;          /* 錯誤，將float型態的指標指向int型態的變數 */
  printf("sizeof(a1)=%d\n",sizeof(a1));
  printf("sizeof(a2)=%d\n",sizeof(a2));
  printf("a1=%d,*ptri=%d\n",a1,*ptri);
  printf("a2=%.1f,*ptrf=%.1f\n",a2,*ptrf);

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

傳遞指標到函數

- 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 */
}
```

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

傳遞指標到函數

- 傳遞指標的應用：

```
#include <stdio.h>
#include <stdlib.h>
void add10(int *);                          /* add10()函數的原型 */
int main(void) {
   int a=5;
   printf("Before call add10(),a=%d\n",a);
   add10(&a);                               /* 呼叫add10()函數 */
   printf("After call add10(),a=%d\n",a);
   system("pause");
   return 0;
}

void add10(int *p1)
{
   *p1=*p1+10;
}
```

變數值的互換

```
#include <stdio.h>
#include <stdlib.h>
void swap(int,int);      /* swap()函數的原型 */
int main(void){
   int a=5,b=20;
   printf("Before swap...");
   printf("a=%d,b=%d\n",a,b);
   swap(a,b);
   printf("After swap...");
   printf("a=%d,b=%d\n",a,b);
   system("pause");
   return 0;
}

void swap(int x,int y)   /* 定義swap()函數 */
{
   int tmp=x;
   x=y;
   y=tmp;
}
```

變數值的互換

```
#include <stdio.h>
#include <stdlib.h>
void swap(int *,int *);   /* 函數swap()原型的宣告 */
int main(void)  {
   int a=5,b=20;
   printf("Before swap...");
   printf("a=%d,b=%d\n",a,b);
   swap(&a,&b);
   printf("After swap...");
   printf("a=%d,b=%d\n",a,b);
   system("pause");
   return 0;
}

void swap(int *p1,int *p2)
{
   int tmp=*p1;
   *p1=*p2;
   *p2=tmp;
}
```

傳回多個數值的函數

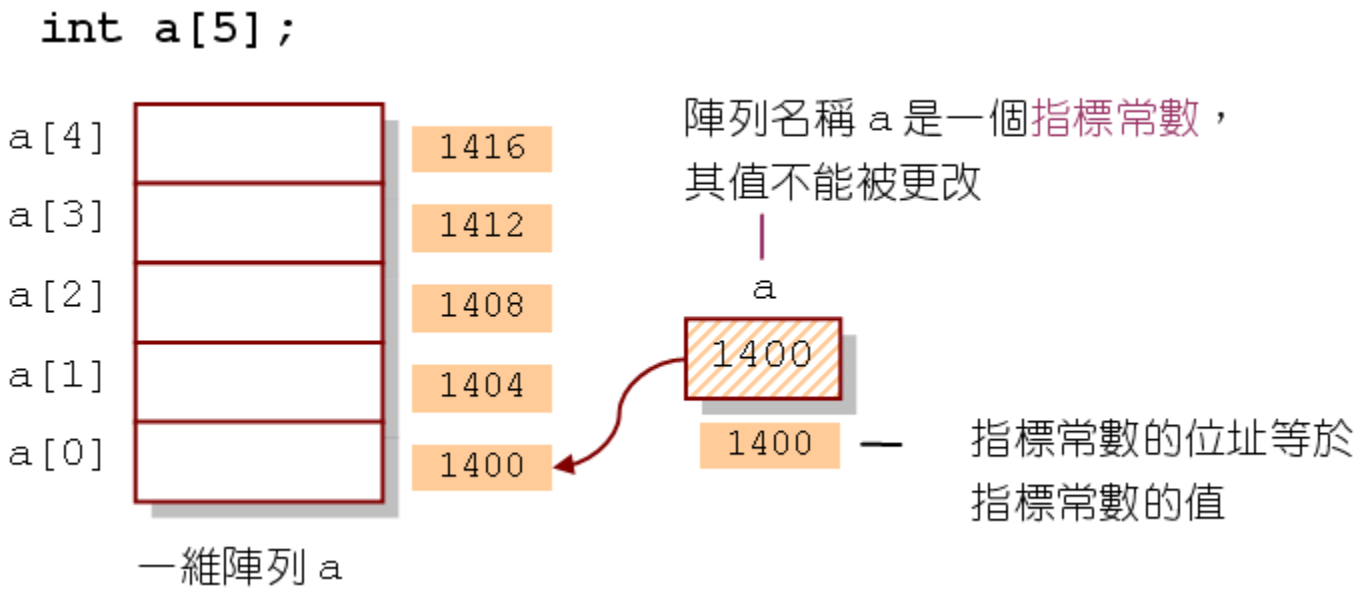
```
#include <stdio.h>
#include <stdlib.h>
void rect(int,int,int *,int *);    /* 函數rect()的原型 */
int main(void)
{
  int a=5,b=8;
  int area,peri;
  rect(a,b,&area,&peri);          /* 呼叫rect(),計算面積及周長 */
  printf("area=%d,total length=%d\n",area,peri);

  system("pause");
  return 0;
}

void rect(int x,int y,int *p1, int *p2)
{
  *p1=x*y;
  *p2=2*(x+y);
}
```

指標與陣列

- 陣列的名稱是一個指標常數，它指向該陣列的位址



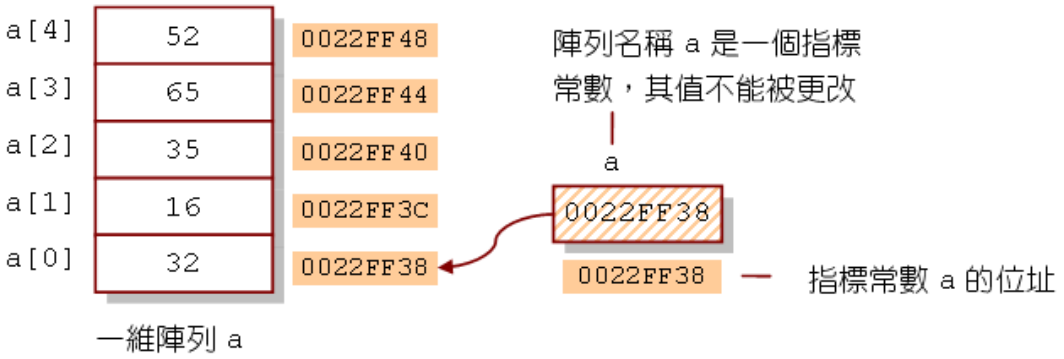
陣列名稱的值即陣列的位址

- 驗證陣列名稱是一個指標常數：

```
#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]);  /* 印出陣列a每一個元素的位址 */

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

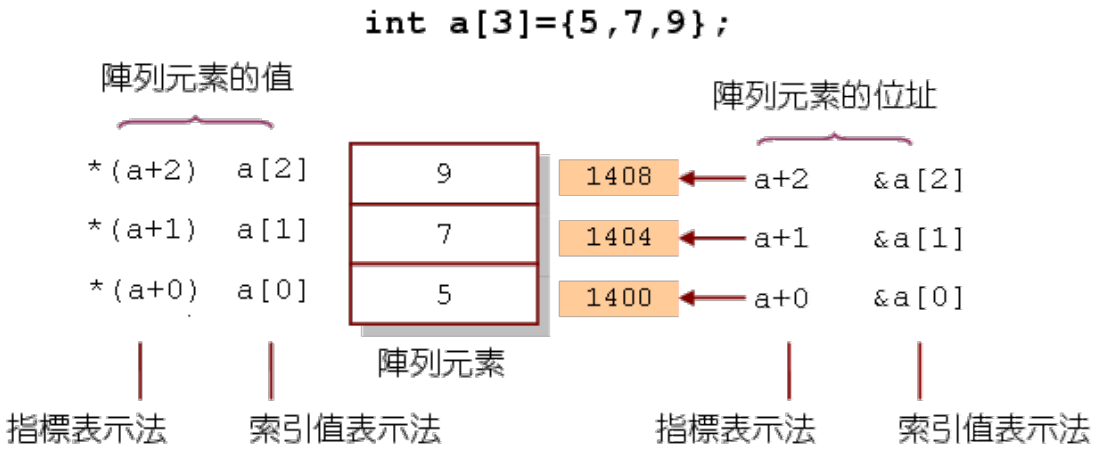


指標的算數運算

- 利用指標存取陣列的內容

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  int a[3]={5,7,9};
  printf("a[0]=%d, *(a+0)=%d\n",a[0],*(a+0));
  printf("a[1]=%d, *(a+1)=%d\n",a[1],*(a+1));
  printf("a[2]=%d, *(a+2)=%d\n",a[2],*(a+2));

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



-The End-