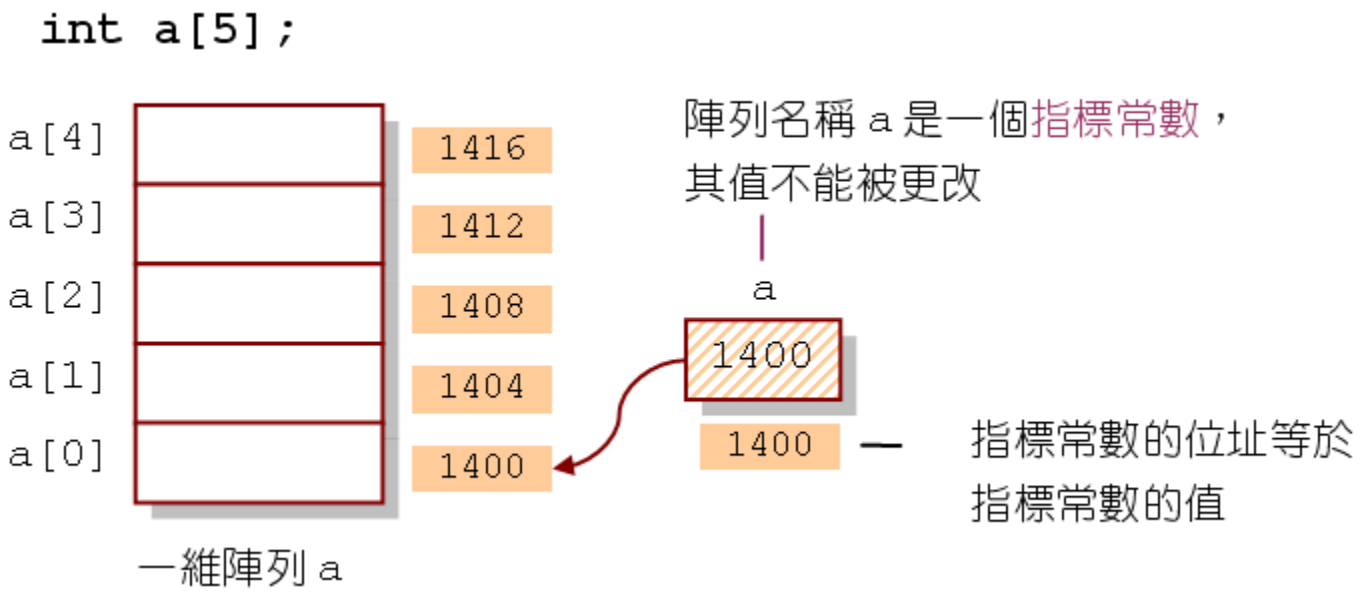
C 程式語言 8

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



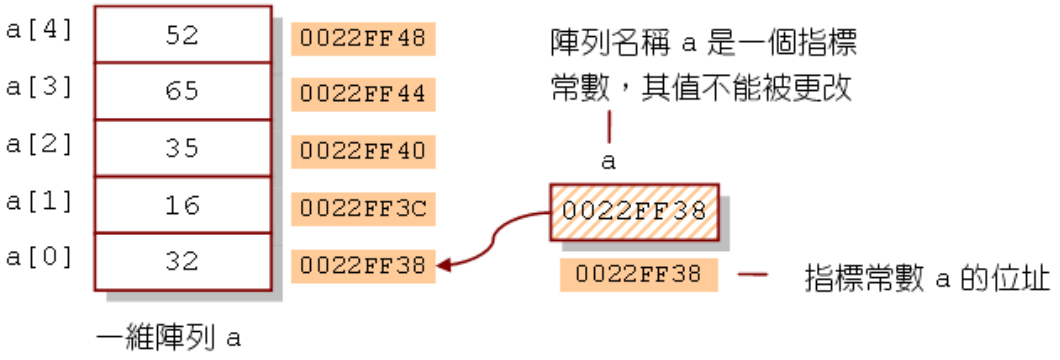
陣列名稱的值即陣列的位址

- 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;
}
```

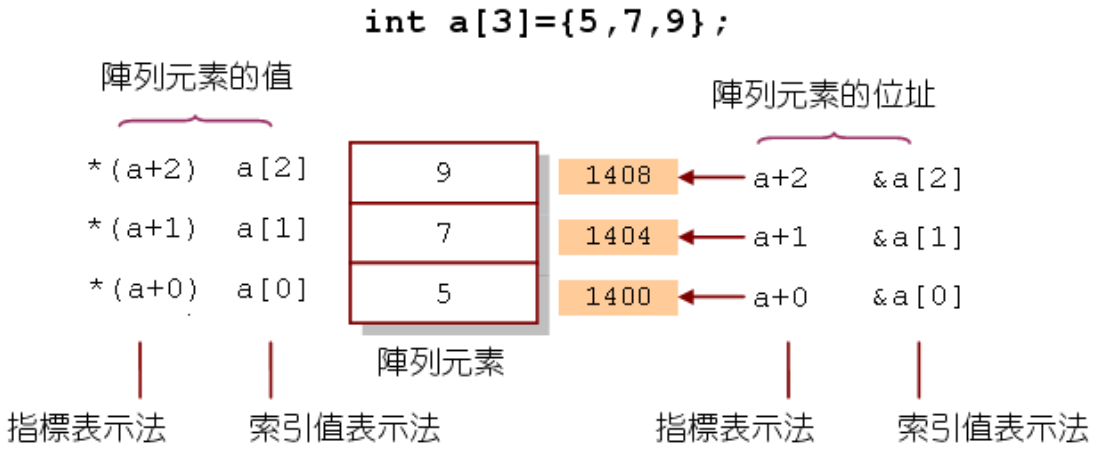


指標的算數運算

- 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;
}
```



- 雙重指標，就是指向指標的指標；
- 雙重指標則需用到**兩個***，亦即需要二次的間接存取才能得到某一變數
  的值。
- 雙重指標型態的定義方式：

  變數型態 **指標變數名稱;

  例：
  - int **ptri;
  - char **ptrc;

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

void ex8_1(void)
{
	int x = 100;	//定義變數 x
	int *p1  = &x;	//定義指標變數
	int **p2 = &p1;	//定義雙重指標變數
	printf("&x=%p, x=%d\n", &x, x);			/* 印出x變數的位址與值 */
	printf("&p1=%p\, p1=%p\n",   &p1, p1);	/* 印出p1指標變數位址 */
	printf("&p2=%p\, p2=%p\n",   &p2, p2);	/* 印出p2指標變數位址 */
	printf("*p1=%d\, **p2=%d\n", *p1, **p2);	/* 印出p1指標變數的值與p2指標變數的值 */
	printf("p1佔%dbytes\, p2佔%dbytes\n", sizeof(p1), sizeof(p2)); /* 印出p1與p2指標變數佔多少bytes位址 */
	system("pause");
	return 0;
}
```

- 雙重指標存放指標變數的位址

除錯: 以下程式, 請幫忙 debug 一下:

1.

```
#include <stdio.h>
int main( )
{
   int a=100;
   int *p = a;
    int **pp = p;
    printf("a=%d, *p=%d, and **pp=%d\n", a, *p, **pp);
    return 0;
}
```

2. 輸入 123.456 作測試

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{
   double d;
   double *p = &d;
    double **pp = p;
    printf("請輸入一 double 數: ");
    scanf("%f", d);
    printf("a=%d, *p=%d, and **pp=%d\n", d, *p, **pp);
    system("PAUSE");
    return 0;
}
```

練習: 修改 ex8_1.c 程式, 將變數的資料型態更改為字元型態, 判斷雙重指標變數占多少 bytes。

ex8_7.c

指標陣列與二維陣列

單一指標與一維陣列的關係

int a[] = {10, 20, 30, 40, 50};
int *ptr = a;

*ptr 與 a[0] 皆表示 10 (陣列第一個元素的整數值), 由此可知, 一維陣列只要使用一個 * 或 [] 就可以得到變數的值, 這表示 * 與 [] 可以互用。

二維陣列與指標的關係:

當資料型態為: 字元型態時

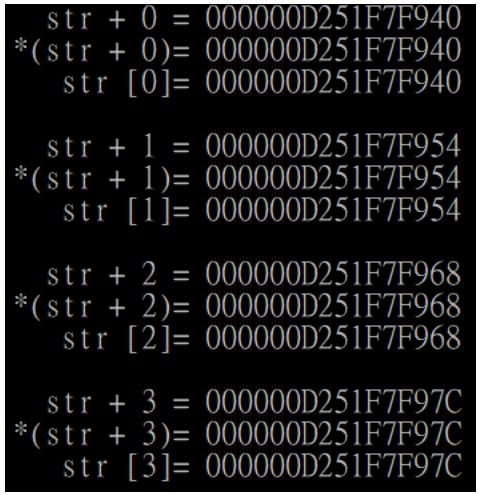
char str[4][20] = { “Department”, “of”, “Information”, “Management” };

str	D	e	p	a	r	t	m	e	n	t	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0
str+1	o	f	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0
str+2	I	n	f	o	r	m	a	t	i	o	n	\0	\0	\0	\0	\0	\0	\0	\0	\0
str+3	M	a	n	a	g	e	m	e	n	t	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0

str 與 str[0] 皆為 “Department” 字串中的 ‘D’ 字元的位址, 即 str 與 str[0] 都表示 &str[0][0]

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{
  int i;
  char str[4][20] = {"Department", "of", "Information", "Management"};
  for(i = 0; i<4; i++){
            printf("str + %d = %p\n", i, str+i);
            printf("*(str + %d) = %p\n", i, *(str+i));
            printf("str [%d] = %p\n\n", i, str[i]);
   }
   system("PAUSE");
   return 0;
}
```

str+i 、*(str+i) 、str[i] ，此處 i = 0、1、2、3 。 所得到的結果是相同的。
因為二維陣列須經由 2 個 * 、2 個 [] 、或一個 [] 及一個 * 才能得到變數的值。



證明:

由一維得知 * 與 [] 可以互用。

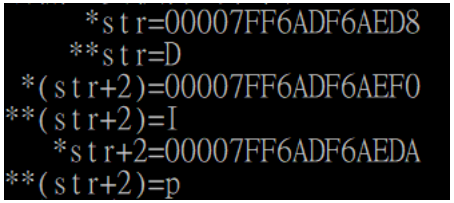
所以:
2 維陣列
char str[4][20] = { “Department”, “of”, “Information”, “Management” };

可以用 陣列指標表示 :
char *str[4] = { “Department”, “of”, “Information”, “Management” };

2 者差異: 以 2 維陣列表示, 較浪費空間且較沒彈性, 陣列指標方式較彈性配置空間, 從而節省空間
因為 [] 的運算優先順序比 * 高, 故:

str 是一個含有 4 個元素的陣列, 每一個元素皆為指向 char 的指標:

```c
#include <stdio.h>
#include <stdlib.h>
int main( )
{
   char *str[4] = { "Department", "of", "Information",
                    "Management" };
   printf("*str=%p\n", *str);
   printf("**str=%c\n", **str);
   printf("*(str+2)=%p\n", *(str + 2));
   printf("**(str+2)=%c\n", **(str + 2));
   printf("*str+2=%p\n", *str + 2);
   printf("**(str+2)=%c\n", *(*str + 2));
   system("PAUSE");
   return 0;
}
```



當資料型態為: 整數型態時

int array[2][3]

- 由 2 個一維陣列所組成，每個一維陣列裡各有3個元素

int arr[2][3] = {{10, 20, 30}, {40, 50, 60}};

以一般二維陣列表示時:

arr	arr[0][0] 10	arr[0][1] 20	arr[0][2] 30
arr+1	arr[0][0] 10	arr[0][0] 10	arr[0][0] 10

arr 和 arr[0] 都表示 arr[0][0] 的位址, arr 是陣列名稱
arr+1 代表第二列第一個元素位址, 即 arr[1][0] 的位址。
arr+2 代表第三列第一個元素位址, 即 arr[2][0] 的位址。
arr+3 代表第四列第一個元素位址, 即 arr[3][0] 的位址。

取元素的值只要使用 arr[0][0]、arr[0][1]、arr[0][2]、arr[1][0]、arr[1][1]、arr[1][2] 就可以得到。

改用陣列指標方法:

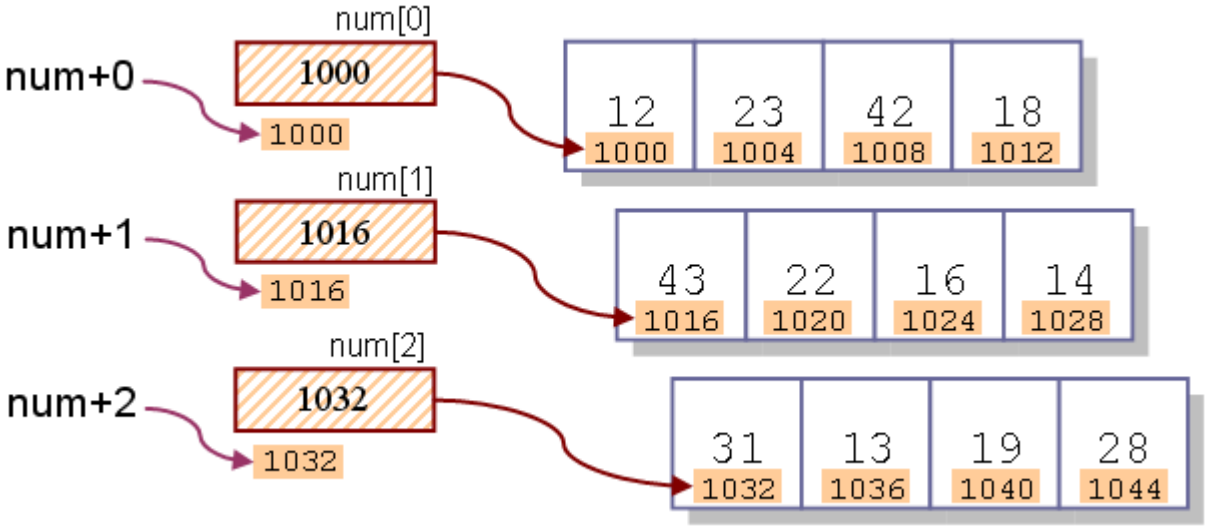
```
int  arr[2][3]  = {{10, 20, 30}, {40, 50, 60}};
int   *ptr[2] = {arr[0], arr[1]};
```

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int arr[2][3] = {{10, 20, 30}, {40, 50, 60}};
   int *ptr[2]={arr[0], arr[1]};
   printf("*ptr=%p\n", *ptr);
   printf("**ptr=%d\n", **ptr);
   printf("*(ptr+1)=%p\n", *(ptr+1));
   printf("**(ptr+1)=%d\n", **(ptr+1));

   printf("*ptr+2=%p\n", *ptr+2);
   printf("*(*ptr+2)=%d\n", *(*ptr+2));
    printf("*(ptr+1)+2=%p\n", *(ptr+1)+2);
    printf("*(*(ptr+1)+2)=%d\n", *(*(ptr+1)+2));
   system("PAUSE");
    return 0;
}
```

```
          *ptr=0000004642B8F578
         **ptr=10
      *(ptr+1)=0000004642B8F584
     **(ptr+1)=40
        *ptr+2=0000004642B8F580
     *(*ptr+2)=30
    *(ptr+1)+2=0000004642B8F58C
*(*(ptr+1)+2)=60
```

二維陣列的指標表示方式



num+i 的值等於 num[i] 的值

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

認識結構

- 「結構」（Structures）和聯合、列舉都屬於自訂資料型態（User-Defined Types），可以讓程式設計者自行在程式碼定義新的資料型態。

- 結構是由一或多個不同資料型態（當然也可以是相同資料型態）所組成的集合，然後使用一個新名稱來代表，新名稱就是一個新的資料型態，我們可以使用此新資料型態來宣告結構變數。

結構宣告與基本使用

- 在C程式宣告結構是使用struct關鍵字來定義新的資料型態，這是一個範本，其語法如下所示：

```
struct 結構名稱 {
    資料型態 成員1;
    資料型態 成員2;
    ……
};
```

- 上述語法定義名為結構名稱的新資料型態，在結構中宣告的變數稱為該結構的「成員」（Members）。

結構宣告與基本使用 – 宣告結構變數

- 在宣告student結構的自訂資料型態後，我們可以在程式碼使用此型態來宣告結構變數，也就是配置記憶體空間來建立「結構實例」（Structure Instance），其語法如下所示：

  struct 結構名稱 變數名稱;

- 上述宣告是使用struct關鍵字開頭加上結構名稱來宣告結構變數，例如：宣告student結構變數，如下所示：

  struct student std1;

認識結構

- 結構可將型態不同的資料合併成為新的型態
- 定義結構與宣告結構變數的格式如下：

定義結構與宣告結構變數的語法

```
struct 結構名稱
{
   資料型態 成員名稱 1;
   資料型態 成員名稱 2;
                  ...
   資料型態 成員名稱 n;
};

struct 結構名稱 變數 1, 變數 2,…, 變數 n;
```

認識結構

- 結構定義的範例

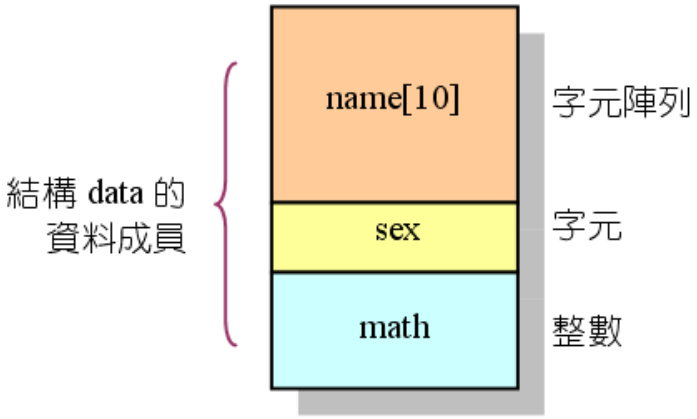
```
struct data      /* 定義 data 結構*/
{
   char name[10];
   char sex;
   int math;
};
struct data mary, tom;   /* 宣告 data 型態的結構變數 */
```

定義結構的成員

定義完結構之後，立即宣告結構變數

```
struct data      /* 定義 data 結構*/
{
   char name[10];
   char sex;
   int math;
}mary,tom;    /* 宣告結構變數 mary 與 tom */
```

認識結構



認識結構

- 存取結構變數的成員：

存取結構變數的成員

結構變數名稱.成員名稱;

```
mary.sex='F';          /* 設定 sex 成員為'F' */
mary.math=95;          /* 設定 math 成員為 95 */
```

使用結構的範例

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  struct data                    /* 定義結構data */
  {
    char name[10];
    int math;
  } student;                     /* 宣告data型態的結構變數student */
  printf("Please type in your name:");
  gets(student.name);
  printf("Please input a score:");
  scanf("%d",&student.math);          /* 輸入學生成績 */
  printf("Name:%s\n", student.name);
  printf("Score:%d\n", student.math);
  system("pause");
  return 0;
}
```

結構變數所佔的記憶空間

- 利用sizeof() 求出結構所佔用的記憶體空間：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  struct data          /* 定義結構 */
  {
    char name[10];
    int math;
  } student;
  printf("sizeof(student)=%d\n",sizeof(student));

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

結構變數初值的設定

要設定結構變數的初值，可利用下面的語法：

```
struct data       /* 定義結構 data */
{
   char name[10];
   int math;
};
struct data student={"Jenny",78};    /* 設定結構變數的初值 */
```

將結構的定義與變數初值的設定合在一起：

```
struct data       /* 定義結構 data */
{
   char name[10];
   int math;
} student={"Jenny",78};    /* 宣告結構變數,並設定初值 */
```

結構變數初值的設定

- 設定結構變數初值的範例

```
#include <stdio.h>
#include <stdlib.h>
int main(void){
  struct data              /* 定義結構data */
  {
    char name[10];
    int math;
  };
  struct data student={"Mary Wang",74};  /* 設定結構變數初值 */

  printf("Name: %s\n",student.name);
  printf(“Score: %d\n",student.math);

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

-The End-