

陣列與字串

陣列

1

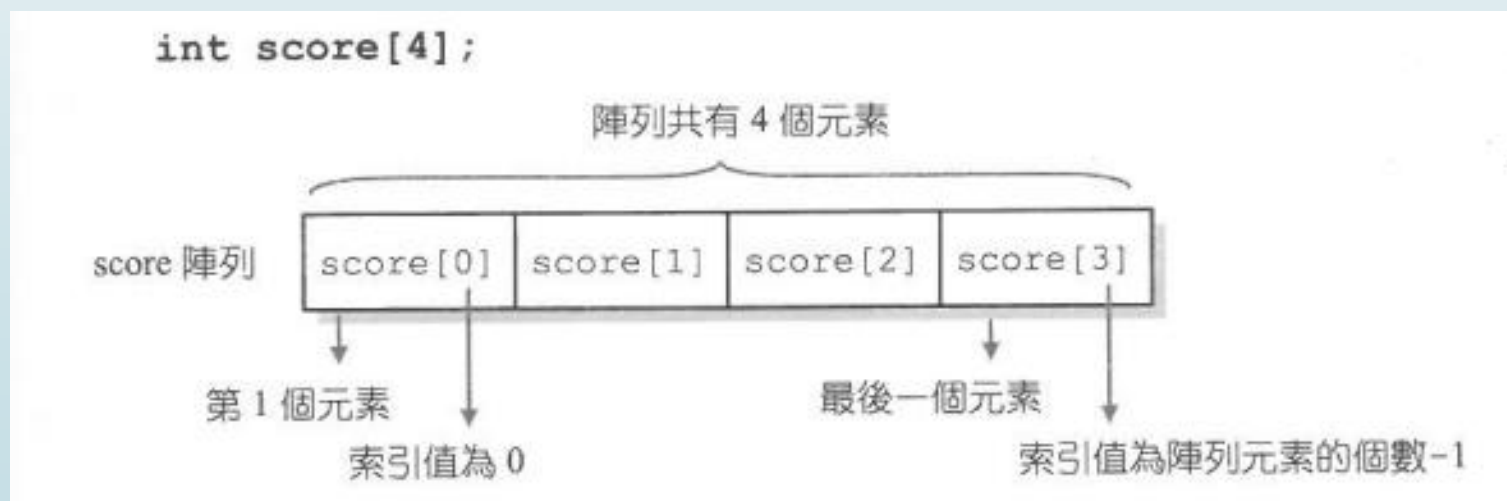
■ 陣列(array)

- 由一群相同型態的變數所組成的一種資料結構，它們以一個共同的名稱表示
- 陣列中個別的元素(element)是以「索引值」(index)來標示存放的位置
- 依照複雜程度分為一維、二維與多維

一維陣列

- 一維陣列(1-dimensional array)
 - 存放多個相同型態的資料
 - 跟變數一樣要先宣告才可以使用
 - 宣告編譯器分配記憶體是一個連續的區塊
- 存取陣列中的元素需要使用索引值(index)
 - Index從0開始
- 宣告格式

資料型態 陣列名稱[個數];



範例-一維陣列

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i,score[4]; /* 宣告整數變數i與整數陣列score */

    score[0]=78;    /* 設定陣列的第一個元素為78 */
    score[1]=55;    /* 設定陣列的第二個元素為55 */
    score[2]=92;    /* 設定陣列的第三個元素為92 */
    score[3]=80;    /* 設定陣列的最後一個元素為80 */

    for(i=0;i<=3;i++)
        printf("score[%d]=%d\n",i,score[i]); /* 印出陣列的內容 */

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

陣列注意事項

- 陣列的元素如果沒有設值，則該元素的值會是原先留在記憶體內的殘值
- C語言不會自動做陣列的索引值界限的檢查，如果陣列索引值超出了原先陣列所宣告所能儲存的範圍時，將會得到無法預期的結果
 - 因為是原先留存於記憶體中的殘值，因此在不同的執行環境裡可能會得到不同結果

範例-陣列注意事項

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i,score[4]; /* 宣告整數變數i與整數陣列score */

    score[0]=78;
    score[1]=55;
    /* score[2]=92; 此行刻意不將score[2]設值 */
    score[3]=80;

    for(i=0;i<=4;i++) /* 此行刻意將索引值超出陣列score的可容許範圍 */
        printf("score[%d]=%d\n",i,score[i]);

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

陣列初值的設定

➡ 格式

資料型態 陣列名稱 [個數 n] = {初值 1, 初值 2, ..., 初值 n};

- ➡ 如果想要將陣列內所有的元素皆設值成同一個值時，只需要在左大括號與又大括號中填入一個數值即可，不管陣列元素多少都會被設成相同值

```
int data[5]={0};    /* 將陣列 data 內的所有元素值都設為 0 */
```

- ➡ 若是在宣告時沒有宣告陣列的個數大小，編譯器則會依據初值個數來決定陣列的大小

```
int score[]={60,75,48,92};    /* 有 4 個初值，所以陣列 score 的大小為 4 */
```

- ➡ 當宣告的陣列大小與實際的初值個數不相同時
 - ➡ 如果初值比宣告陣列小時，則剩餘未設值的空間會填入0
 - ➡ 如果初值比宣告陣列大時，編譯器則會出現警告訊息

```
excess elements in array initializer    /* 編譯器的警告訊息 */
```

查詢陣列所佔的記憶空間

■ 查詢陣列所佔的位元組

```
sizeof(陣列名稱) /* 查詢陣列所佔的位元組 */
```

■ C語言沒有提供查詢陣列大小的函數

■ 取出陣列大小的方法

- 利用陣列所佔的位元組去除以陣列的資料型態，即為陣列的個數

範例-取出陣列的大小

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    double data[4]; /* 宣告有4個元素的double型態陣列 */
    printf("陣列元素所佔的位元組:%d\n", sizeof(data[0]));
    printf("整個陣列所佔的位元組:%d\n", sizeof(data));
    printf("陣列元素的個數:%d\n", sizeof(data)/sizeof(double));

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

範例-陣列初值由鍵盤輸入

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i, age[3];
    for(i=0; i<3; i++)
    {
        printf("請輸入age[%d]的值:", i);
        scanf("%d", &age[i]);    /* 由鍵盤輸入數值給陣列age裡的元素 */
    }
    for(i=0; i<3; i++)
        printf("age[%d]=%d\n", i, age[i]);

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

範例-陣列最大值及最小值

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int A[5]={74,48,30,17,62};
    int i,min,max;
    min=max=A[0];          /* 將max與min均設為陣列的第一個元素 */

    for(i=0;i<5;i++)
    {
        if(A[i]>max) /* 判斷A[i]是否大於max */
            max=A[i];
        if(A[i]<min) /* 判斷A[i]是否小於min */
            min=A[i];
    }
    printf("陣列裡元素的最大值為%d\n",max);
    printf("陣列裡元素的最小值為%d\n",min);

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

範例-輸入未知個數的資料到陣列

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10      /* 定義MAX為10 */
int main(void)
{
    int score[MAX]; /* 宣告有10個元素的整數陣列 */
    int i=0,num;
    int sum=0;      /* 宣告用來成績總和的變數sum */

    printf("請輸入成績，要結束請輸入0:\n");
    do
    {
        printf("請輸入成績:");
        scanf("%d",&score[i]);
    }while(score[i++]>0); /* 輸入成績，輸入0時結束 */
    num=i-1;
    for(i=0;i<num;i++)
        sum+=score[i]; /* 計算平均成績 */
    printf("平均成績為 %.2f\n",(float)sum/num);
    system("pause");
    return 0;
}
```

陣列界限的檢查

- C語言不會檢查陣列索引值的大小，當索引值超過陣列的長度時，C語言不會不讓使用者繼續使用該陣列，它只是會將多餘的資料放在陣列之外的記憶體中，如此一來很可能會蓋掉其他的資料或是程式碼，因此產生不可預期的錯誤，這種錯誤是執行時才會發生(run-time error)，而不是編譯時發生的錯誤(compiler-time error)，因此編譯程式不會有任何的警告訊息
- 由於C語言為了增加執行的速度，不會做陣列界限的檢查，而是交由程式設計師，為了避免發生不可預期的錯誤，需要在程式中加入陣列界限的檢查

範例 – 陣列的界限檢查

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5      /* 定義MAX為5 */
int main(void)
{
    int score[MAX]; /* 宣告score陣列，可存放MAX個整數 */
    int i=0,num;
    float sum=0.0f;

    printf("請輸入成績，要結束請輸入0:\n");
    do
    {
        if(i==MAX)      /* 當i的值為MAX時，表示陣列已滿，即停止輸入 */
        {
            printf("陣列空間已使用完畢!!\n");
            i++;          /* 此行先將i值加1，因為23行會把i的值減1掉 */
            break;
        }
        printf("請輸入成績:");
        scanf("%d",&score[i]);
    }while(score[i++]>0); /* 輸入成績，輸入0時結束 */
    num=i-1;
    for(i=0;i<num;i++)
    {
        sum+=score[i];    /* 計算平均成績 */
    }
    printf("平均成績為 %.2f\n",sum/num);

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

範例－陣列的搜尋

```
/* prog9_8, 陣列的搜尋 */
#include <stdio.h>
#include <stdlib.h>
#define SIZE 6      /* 定義SIZE為6 */
int main(void)
{
    int i,num,flag=0;
    int A[SIZE]={33,75,69,41,33,19};

    printf("陣列A元素的值為:");
    for(i=0;i<SIZE;i++)
        printf("%d ",A[i]);          /* 印出陣列的內容 */

    printf("\n請輸入欲搜尋的整數:");
    scanf("%d",&num);                /* 輸入欲搜尋的整數 */

    for(i=0;i<SIZE;i++)
        if(A[i]==num) /* 判斷陣列元素是否與輸入值相同 */
        {
            printf("找到了! A[%d]=%d\n",i,A[i]);
            flag=1;          /* 設flag為1，代表有找到相同的數值 */
        }
    if(flag==0)
        printf("沒有找到相同值!!\n");

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

二維陣列

■ 二維陣列(2-dimensional array)

- 和一維陣列類似
- 加入列與行的關係
- 宣告

資料型態 陣列名稱 [列的個數] [行的個數];

■ 範例

```
int data[10][5];    /* 宣告整數陣列 data，可存放 10 列 5 行的整數資料 */  
float score[4][3]; /* 宣告浮點數陣列 score，可存放 4 列 3 行的浮點數資料 */
```

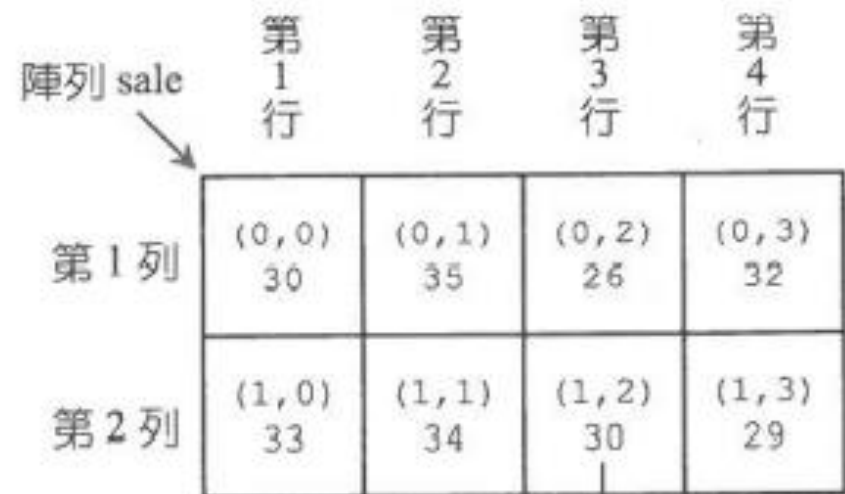
■ 宣告初值語法

資料型態 陣列名稱 [列的個數] [行的個數] = { { 第 1 列初值 },
{ 第 2 列初值 },
{ ... },
{ 第 n 列初值 } };

二維陣列存放

表 9.2.1 業務員於 2004 年每季的銷售業績

業務員	2004 年銷售量			
	第一季	第二季	第三季	第四季
1	30	35	26	32
2	33	34	30	29



每一格代表一個元素，每個元素皆為 int 型態

二維陣列宣告

```
int sale[2][4]={ {30,35,26,32},  
                 {33,34,30,29}};
```

2×4的陣列是由2個具有4個元素的一維陣列所組成

```
int sale[2][4]={ {30,35,26,32}, {33,34,30,29}};
```

2×4 的陣列

一維陣列，
有 4 個元素

一維陣列，
有 4 個元素

二維陣列宣告

- C語言允許二維與二維以上的多維陣列在設定初值時，可以省略第一個索引值，但其他則必須填寫，可方便的增加或減少陣列的大小

```
int temp[][4]={ {30,35,26,32},  
                {33,34,30,29},  
                {25,33,29,25}};
```

範例 - 二維陣列元素的存取

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i,j,sale[2][4],sum=0;

    for(i=0;i<2;i++)
        for(j=0;j<4;j++)
        {
            printf("業務員%d的第%d季業績:",i+1,j+1);
            scanf("%d",&sale[i][j]);    /* 輸入銷售量 */
        }

    printf("***Output***");
    for(i=0;i<2;i++)    /* 輸出銷售量並計算總銷售量 */
    {
        printf("\n業務員%d的業績分別為",i+1);
        for(j=0;j<4;j++)
        {
            printf("%d ",sale[i][j]);
            sum+=sale[i][j];
        }
    }
    printf("\n2004年總銷售量為%d部車\n",sum);

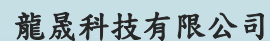
    system("pause");
    return 0;
}
```

範例 - 矩陣的相加

```
#include <stdio.h>
#include <stdlib.h>
#define ROW 2      /* 定義ROW為2 */
#define COL 3      /* 定義COL為3 */
int main(void)
{
    int i,j;
    int A[ROW][COL]={1,2,3},{5,6,8}; /* 宣告陣列A並設定初值 */
    int B[ROW][COL]={3,0,2},{3,5,7}; /* 宣告陣列B並設定初值 */

    printf("Matrix A+B=\n");
    for(i=0;i<ROW;i++)                /* 外層迴圈，用來控制列數 */
    {
        for(j=0;j<COL;j++)            /* 內層迴圈，用來控制行數 */
            printf("%3d",A[i][j]+B[i][j]); /* 計算二陣列相加 */
        printf("\n");
    }
    system("pause");
    return 0;
}
```

```
int A[2][4][3];    /* 宣告 2×4×3 整數陣列 A */
```



範例-三維陣列中的最大值

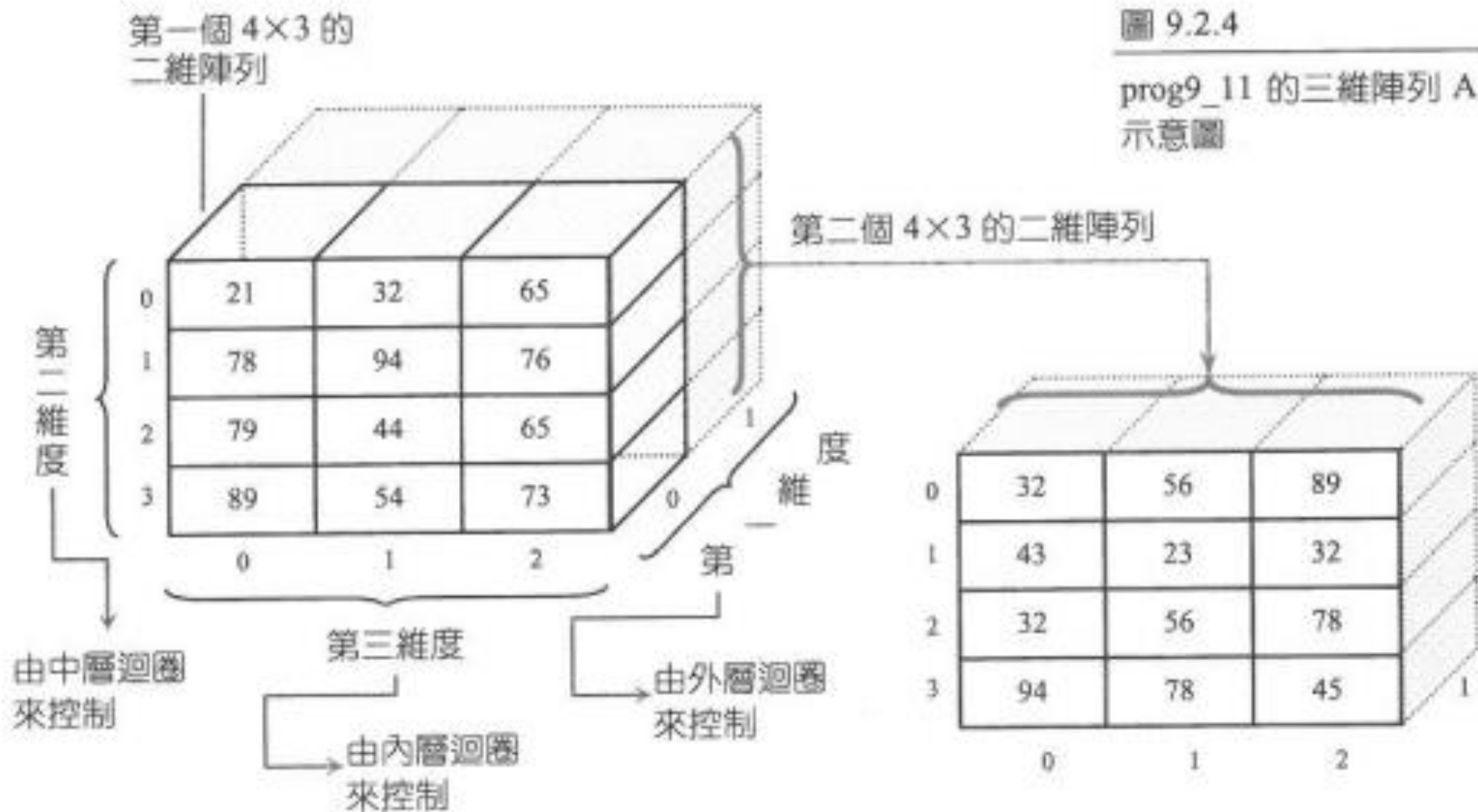
```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int A[2][4][3]={{21,32,65},
                    {78,94,76},
                    {79,44,65},
                    {89,54,73}},
        {{32,56,89},
        {43,23,32},
        {32,56,78},
        {94,78,45}}};

    int i,j,k,max=A[0][0][0];    /* 設定max為A陣列的第一個元素 */

    for(i=0;i<2;i++)            /* 外層迴圈 */
        for(j=0;j<4;j++)        /* 中層迴圈 */
            for(k=0;k<3;k++)     /* 內層迴圈 */
                if(max<A[i][j][k])
                    max=A[i][j][k];

    printf("max=%d\n",max);    /* 印出陣列的最大值 */
    system("pause");
    return 0;
}
```

三維陣列示意圖



將圖解轉換程式碼

4×3 的二維陣列為

```
{ {21, 32, 65},  
  {78, 94, 76},  
  {79, 44, 65},  
  {89, 54, 73} }
```

第二個 4×3 的二維陣列為

```
{ {32, 56, 89},  
  {43, 23, 32},  
  {32, 56, 78},  
  {94, 78, 45} }
```

```
int A[2][4][3] = { { {21, 32, 65},  
                     {78, 94, 76},  
                     {79, 44, 65},  
                     {89, 54, 73} },  
                   { {32, 56, 89},  
                     {43, 23, 32},  
                     {32, 56, 78},  
                     {94, 78, 45} } }
```

第一個 4×3 的二維陣列

第二個 4×3 的二維陣列

2×4×3 的三維陣列

傳遞陣列給函數

- C語言在傳遞陣列給函數時，並不是一整個陣列，而是傳遞存放陣列的記憶體位址，函數裡的程式碼便是根據陣列的位址來進行元素的處理

- 宣告

```
傳返回值型態 函數名稱 (資料型態 陣列名稱[]); /* 原型 */  
int main(void)  
{  
    資料型態 陣列名稱 [個數];  
    ...  
    函數名稱 (陣列名稱);  
    ...  
}  
傳返回值型態 函數名稱 (資料型態 陣列名稱 [ ] )  
{  
    ...  
}
```

陣列括號內可以不填
元素的個數

範例 - 傳遞一維陣列到函數

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4
void show(int arr[]);          /* 宣告函數show()的原型 */
int main(void)
{
    int A[SIZE]={5,3,6,1};     /* 設定陣列A的初值 */
    printf("陣列的內容為: ");
    show(A);                   /* 呼叫函數show() */

    system("pause");
    return 0;
}
void show(int arr[])           /* 函數show()的定義 */
{
    int i;
    for(i=0;i<SIZE;i++)
        printf("%d ",arr[i]); /* 印出陣列內容 */
    printf("\n");
}
```

函數傳遞引數的機制

- ➡ 傳值呼叫 (call by value)
 - ➡ 變數型態: int char 等除(陣列外)
 - ➡ 將變數複製一份出來變成區域變數使用
- ➡ 傳址呼叫 (call by address)
 - ➡ 陣列變數
 - ➡ 記憶體位址
 - ➡ 由於陣列的長度可能很大，基於執行效率上的考量，C語言採用該陣列存放於記憶體中的位址

範例 – 印出變數的位址

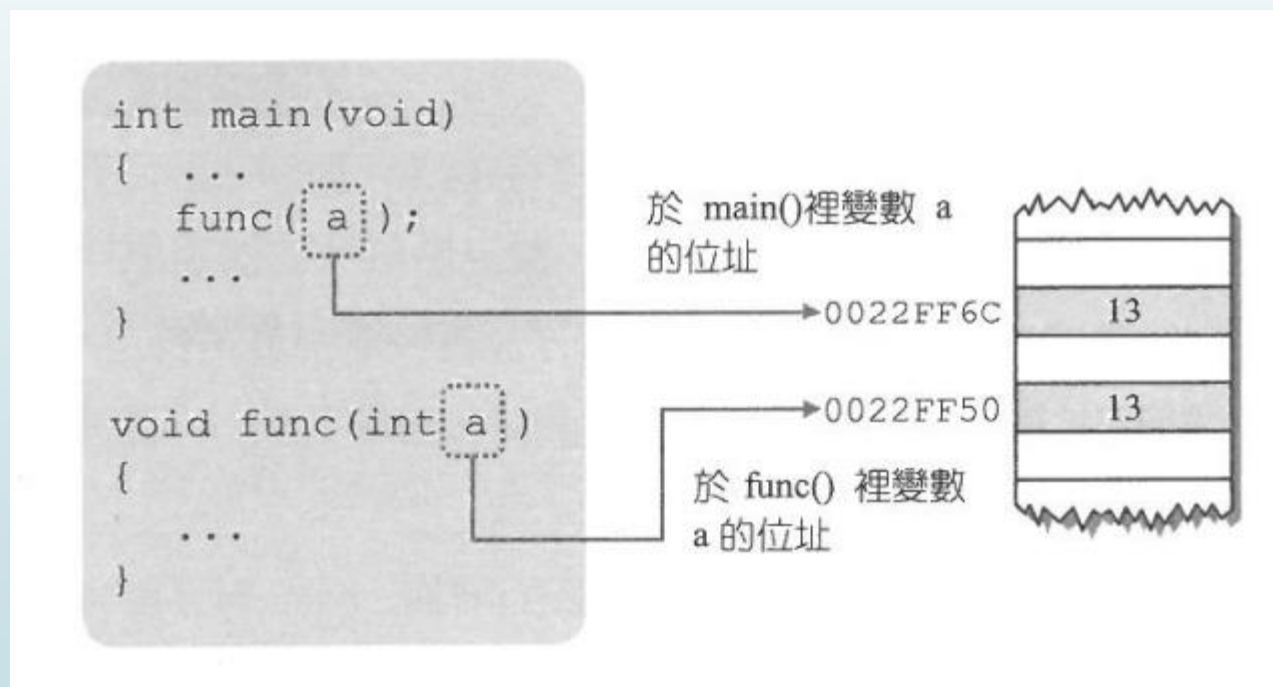
```
#include <stdio.h>
#include <stdlib.h>
void func(int);
int main(void)
{
    int a=13;
    printf("於main()裡,a=%d,a的位址=%p\n",a,&a);
    func(a);          /* 這是傳值呼叫的機制 */

    system("pause");
    return 0;
}

void func(int a)      /* 自訂函數func() */
{
    printf("於func()裡,a=%d,a的位址為=%p\n",a,&a);
}
```

範例解說

- 印出變數位址
 - 變數前加上「&」符號
 - 列印出來時使用「%p」
- 傳值呼叫示意圖



範例 – 印出陣列的位址

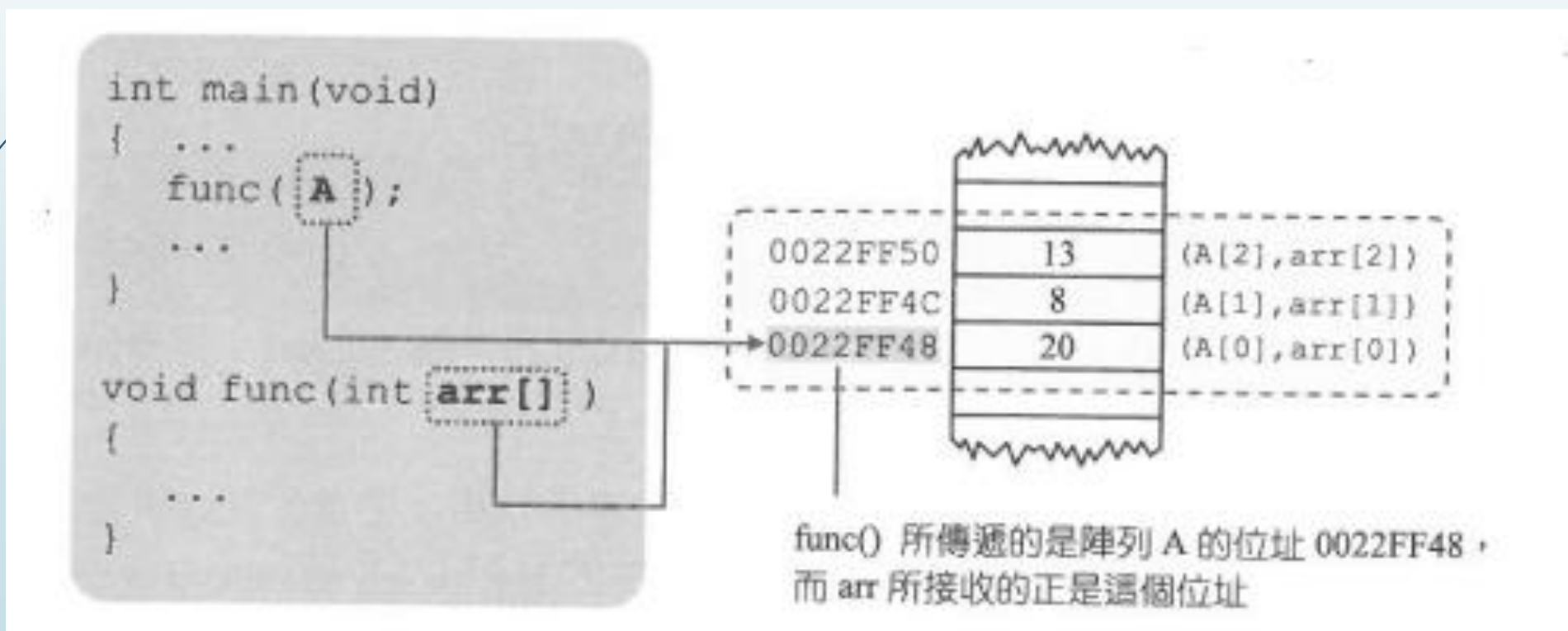
```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 3
void func(int arr[]);
int main(void)
{
    int i,A[SIZE]={20,8,13};

    printf("在main()裡，陣列A元素的位址為\n");
    for(i=0;i<SIZE;i++)
        printf("A[%d]=%2d,位址為%p\n",i,A[i],&A[i]);
    func(A); /* 這是傳址呼叫的機制 */

    system("pause");
    return 0;
}
void func(int arr[]) /* 自訂函數func() */
{
    int i;
    printf("\n在func()裡，陣列arr元素的位址為\n");
    for(i=0;i<SIZE;i++)
        printf("arr[%d]=%2d,位址為%p\n",i,arr[i],&arr[i]);
}
```

範例解說

- 陣列A的每一個元素的位址與func()函數裡的陣列的每一個元素的位址均相同
- 傳址呼叫示意圖



陣列的位址

- C語言是以陣列第一個元素的位址當成是陣列的位址
- 如果是二維陣列，則是第一列第一行的元素位址維陣列的位址
- 陣列名稱本身就是存放陣列位址的變數

範例 – 印出陣列的位址

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 3
int main(void)
{
    int i,A[SIZE]={20,8,13};

    for(i=0;i<SIZE;i++)
        printf("A[%d]=%2d,位址為%p\n",i,A[i],&A[i]);
    printf("陣列A的位址=%p\n",A);    /* 印出陣列A的位址 */

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

範例 - 在函數中更改陣列元素的值

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4
void show(int arr[]);      /* 函數show()的原型 */
void add2(int arr[]);     /* 函數add2()的原型 */

int main(void)
{
    int A[SIZE]={5,3,6,1};
    printf("呼叫add2()前,陣列的內容為: ");
    show(A);              /* 呼叫函數show() */
    add2(A);              /* 呼叫函數add2() */
    printf("呼叫add2()後,陣列的內容為: ");
    show(A);              /* 呼叫函數show() */

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

```
void show(int arr[])
{
    int i;
    for(i=0;i<SIZE;i++) /* 印出陣列內容 */
        printf("%d ",arr[i]);
    printf("\n");
}

void add2(int arr[])
{
    int i;
    for(i=0;i<SIZE;i++)
        arr[i]+=2;
}
```

氣泡排序法

► bubble sort

► 將數值大的數字慢慢往右移動

0		26	5	81	7	63
1		5	26	7	63	81
2		5	7	26	63	81
3		5	7	26	63	81

範例 - 氣泡排序法

```

#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
void show(int a[]), bubble(int a[]);    /* 定義函數的原型 */
int main(void)
{
    int data[SIZE]={26,5,81,7,63};

    printf("排序前...\n");
    show(data);                        /* 印出陣列內容 */
    bubble(data);                      /* 呼叫bubble()函數 */
    printf("排序後...\n");
    show(data);                        /* 印出陣列內容 */
    system("pause");
    return 0;
}

void show(int a[])                    /* 自訂函數show() */
{
    int i;
    for(i=0;i<SIZE;i++)
        printf("%d ",a[i]);          /* 印出陣列的內容 */
    printf("\n");
}

```

```


/* 自訂函數bubble() */
void bubble(int a[])
{
    int i,j,temp;
    for(i=1;i<SIZE;i++)
        for(j=0;j<(SIZE-i);j++)
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
}

```

範例詳解

26	5	81	7	63
----	---	----	---	----

原始陣列

 上色的方塊代表
數字有對換過

第一次搜尋， $i=1$ ， $j=0\sim3$

	a[0]	a[1]	a[2]	a[3]	a[4]
j=0	5	26	81	7	63
j=1	5	26	81	7	63
j=2	5	26	7	81	63
j=3	5	26	7	63	81

執行完 30~35 行 if 敘述之後的結果

第二次搜尋， $i=2$ ， $j=0\sim2$

	a[0]	a[1]	a[2]	a[3]	a[4]
j=0	5	26	7	63	81
j=1	5	7	26	63	81
j=2	5	7	26	63	81

執行完 30~35 行 if 敘述之後的結果

第三次搜尋， $i=3$ ， $j=0\sim1$

	a[0]	a[1]	a[2]	a[3]	a[4]
j=0	5	7	26	63	81
j=1	5	7	26	63	81

執行完 30~35 行 if 敘述之後的結果

第四次搜尋， $i=4$ ， $j=0$

	a[0]	a[1]	a[2]	a[3]	a[4]
j=0	5	7	26	63	81

執行完 30~35 行 if 敘述之後的結果

範例 – 改善氣泡排序法

```

#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
void show(int a[]), bubble2(int a[]);
int main(void)
{
    int data[SIZE]={26,5,81,7,63};

    printf("Before process...\n");
    show(data);
    bubble2(data);
    printf("After process...\n");
    show(data);
    system("pause");
    return 0;
}

void show(int a[])          /* 自訂函數show() */
{
    int i;

    for(i=0; i<SIZE; i++)
        printf("%d ", a[i]);    /* 印出陣列的內容 */
    printf("\n");
}

```

```

/* 氣泡排序函數 */
void bubble2(int a[])
{
    int i, j, temp;
    int flag=0; /* 設定flag為0 */

    for(i=1; (i<SIZE)&&!flag; i++)
    {
        flag=1;
        for(j=0; j<(SIZE-i); j++)
            if(a[j]>a[j+1])
            {
                temp=a[j]; /* 對換陣列內的值 */
                a[j]=a[j+1];
                a[j+1]=temp;
                flag=0;
            }
    }
}

```

範例詳解

第一次搜尋， $i=1$ ， $j=0\sim3$

執行完 33 行之後的結果

a[0]	a[1]	a[2]	a[3]	a[4]	flag
26	5	81	7	63	1

執行完 35~41 行 if 敘述之後的結果

j=0	5	26	81	7	63	0
j=1	5	26	81	7	63	0
j=2	5	26	7	81	63	0
j=3	5	26	7	63	81	0

第二次搜尋， $i=2$ ， $j=0\sim2$

執行完 33 行之後的結果

a[0]	a[1]	a[2]	a[3]	a[4]	flag
5	26	7	63	81	1

執行完 35~41 行 if 敘述之後的結果

j=0	5	26	7	63	81	1
j=1	5	7	26	63	81	0
j=2	5	7	26	63	81	0

第三次搜尋， $i=3$ ， $j=0\sim1$

執行完 33 行之後的結果

a[0]	a[1]	a[2]	a[3]	a[4]	flag
5	7	26	63	81	1

執行完 35~41 行 if 敘述之後的結果

j=0	5	26	7	63	81	1
j=1	5	7	26	63	81	1

因 flag 的值為 1，31 行判斷不成立，故跳離 for 迴圈，結束程式

傳遞二維與多維陣列

語法

```
傳回值型態 函數名稱 (資料型態 陣列名稱[] [行的個數]); /* 原型 */
int main(void)
{
    資料型態 陣列名稱 [列的個數] [行的個數];
    ...
    函數名稱 (陣列名稱);
    ...
}
傳回值型態 函數名稱 (資料型態 陣列名稱[] [行的個數])
{
    ...
}
```

↓
必須填入行的個數

↓
必須填入行的個數

- 不管陣列的維度多少，第一個維度可以不填入元素的個數，後面所有的中括號內都必須填寫數值，為了讓編譯程式能夠處理陣列內各元素的位置

範例 - 尋找二維陣列的最大值與最小值

```
#include <stdio.h>
#include <stdlib.h>
#define ROW 4
#define COL 3
void search(int a[][COL],int b[]);    /* search() 函數的原型 */
int main(void)
{
    int a[ROW][COL]= {{26, 5, 7},
                       {10, 3,47},
                       { 6,76, 8},
                       {40, 4,32}};
    int i,j,b[2];
    printf("二維陣列內的元素:\n");    /* 印出陣列的內容 */
    for(i=0;i<ROW;i++)
    {
        for(j=0;j<COL;j++)
            printf("%02d ",a[i][j]);
        printf("\n");
    }
    search(a,b);                      /* 呼叫search()函數 */
    printf("陣列的最大值=%02d\n",b[0]); /* 印出陣列的最大值 */
    printf("陣列的最小值=%02d\n",b[1]); /* 印出陣列的最小值 */
    system("pause");
    return 0;
}
```

```
void search(int arr[][COL],int p[]) /* 自訂函數search() */
{
    int i,j;
    p[0]=p[1]=arr[0][0];    /* 將p[0]與p[1]均設為arr[0][0] */
    for(i=0;i<ROW;i++)
        for(j=0;j<COL;j++)
        {
            if(p[0]<arr[i][j])    /* 尋找陣列裡的最大值 */
                p[0]=arr[i][j];
            if(p[1]>arr[i][j])    /* 尋找陣列裡的最小值 */
                p[1]=arr[i][j];
        }
}
```

字串

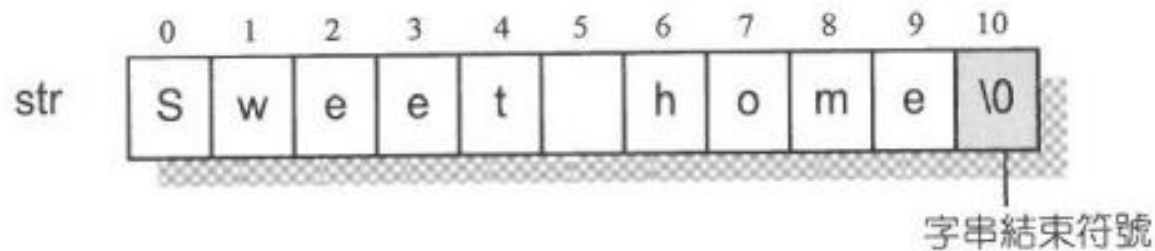
- ➡ C語言沒有「字串」的資料型態，而是由字元陣列來組成字串
- ➡ 字元用「'」字串則是用「"」
- ➡ C語言把字串常數儲存在字元陣列時，會在最後面額外加上字串結束字元「\0」做為結尾，故此宣告字元陣列長度宣告時需n+1的大小

```
char ch_arr[10]={'S','w','e','e','t',' ','h','o','m','e'};
```

```
char str[11]={'S','w','e','e','t',' ','h','o','m','e','\0'};
```

- ➡ 語法 `char 字元陣列名稱[陣列大小] = 字串常數;`

```
char str[11]="Sweet home"; /* 宣告字串變數 str，並設定初值為 Sweet home */
```



範例-印出字元及字串的長度

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char ch='a';          /* 宣告字元變數ch */
    char str1[]="a";      /* 宣告字串變數str1 */
    char str2[]="Sweet home"; /* 宣告字串變數str2 */

    printf("字元ch佔了%d個位元組\n", sizeof(ch));
    printf("字串str1佔了%d個位元組\n", sizeof(str1));
    printf("字串str2佔了%d個位元組\n", sizeof(str2));

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

字串的輸入與輸出函數

- 因 scanf 無法讀取字串裡的空白，故此需要使用到 gets()與 puts() 函數，皆定義在stdio.h裡

- gets (get string)

- 按下「Enter」時才會將該字串接收並加上結束字元「\0」

- 語法 `gets(字元陣列名稱);`

- 字元陣列名稱前面不需要加上位址運算子「&」

- puts (put string)

- 逐一輸出字串，直到遇到字串結束字元「\0」為止，輸出字串後會自動換行

- 語法

```
puts(字元陣列名稱);  
或者是  
puts(字串常數);
```

範例-輸入及印出字串

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char name[15];    /* 宣告字元陣列name */

    puts("What's your name?");
    gets(name);        /* 利用gets()讀入字串，並寫入字元陣列name裡 */
    puts("Hi!");
    puts(name);        /* 印出字元陣列name的內容 */
    puts("How are you?");
    system("pause");
    return 0;
}
```

範例-將字串小寫字母轉大寫

```
#include <stdio.h>
#include <stdlib.h>
void toUpper(char s[]); /* 宣告函數toUpper()的原型 */
int main(void)
{
    char str[15];          /* 宣告可容納15個字元的陣列str */

    printf("請輸入一個字串: ");
    gets(str);             /* 輸入字串 */
    toUpper(str);          /* 呼叫toUpper() 函數 */
    printf("轉換成大寫後: %s\n", str); /* 印出str字串的內容 */

    system("pause");
    return 0;
}

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

字串陣列

➡ 語法 `char 字元陣列名稱[字串的個數][字串長度];`

➡ 設定初始值

```
char 字元陣列名稱[字串的個數][字串長度]=  
{"字串常數 1", "字串常數 2", ..., "字串常數 n"};
```

➡ 範例

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


範例-字串陣列

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    char S[3][10]= {"Tom","Lily","James Lee"};
    int i;

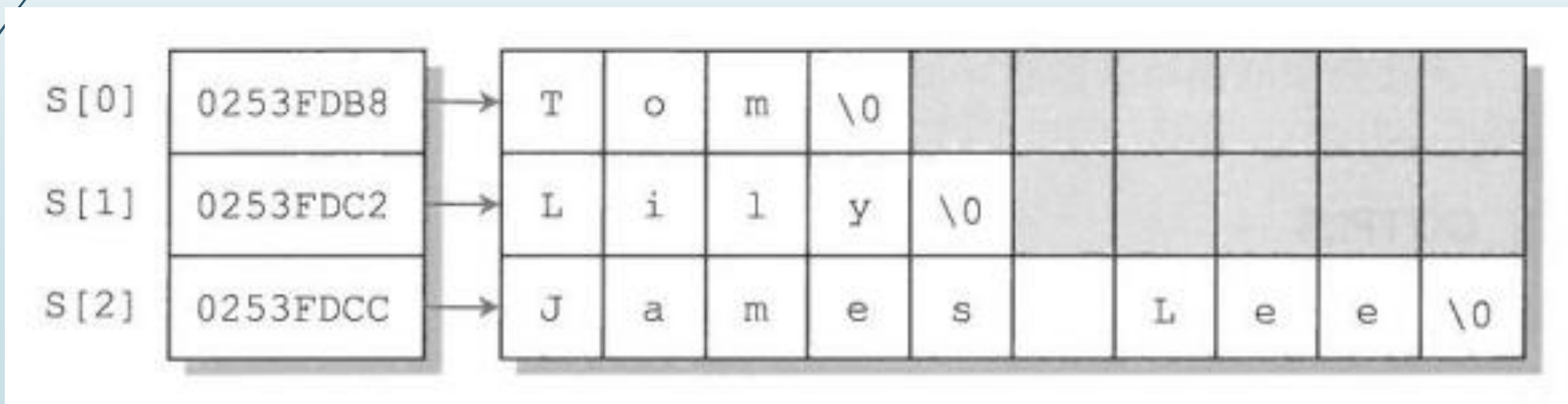
    for(i=0;i<3;i++)
        printf("S[%d]=%s\n",i,S[i]);    /* 印出字串陣列內容 */
    printf("\n");
    for(i=0;i<3;i++) /* 印出字串陣列元素的位址 */
    {
        printf("S[%d]=%p\n",i,S[i]);
        printf("address of S[%d][0]=%p\n\n",i,&S[i][0]);
    }
    system("pause");
    return 0;
}
```

範例解說

► printf

► %s 印出字串

► %p 記憶體位址，陣列名稱本身一個常數，他存放了該帳列所在的記憶體位址



範例-字串陣列的複數

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 3
#define LENGTH 10
int main(void)
{
    char arr1[MAX][LENGTH]={"Tom","Lily","James Lee"};
    char arr2[MAX][LENGTH];
    int i,j;

    for(i=0;i<MAX;i++)    /* 將arr1的內容複製到arr2中 */
    {
        for(j=0;j<LENGTH;j++)
            if(arr1[i][j]=='\0')    /* 如果遇到「\0」,代表讀到字串結束 */
                break;            /* 此行的break敘述會跳到第19行執行 */
            else
                arr2[i][j]=arr1[i][j];
        arr2[i][j]='\0';
    }
    for(i=0;i<MAX;i++)
        printf("arr2[%d]=%s\n",i,arr2[i]);    /* 印出陣列arr2的內容 */
    system("pause");
    return 0;
}
```

習題

51

- 試寫出一程式，找出一維整數陣列元素的最大值的索引值與最小值的索引值

```
arr[] 中的所有值 : 20 0 58 60 -60 78 -92 35 2 1  
arr[] 中最大值的索引值為 5；最小為 6。  
請按任意鍵繼續 . . .
```

```
arr[] 中的所有值 : -90 35 90 100 5 78 87 89 65 25  
arr[] 中最大值的索引值為 3；最小為 0。  
請按任意鍵繼續 . . .
```

習題

52

- 試寫一程式，由鍵盤輸入一字串後，分別計算該字串a、e、i、o、u字母的次數

```
Enter a string : i love c language  
This string have 2 'a', 2 'e', 1 'i', 1 'o' ,and 1 'u'.  
請按任意鍵繼續 . . .
```

習題

53

- 試寫一函數 `void reverse(char str[])`，它可將字串 `str` 反序列印出來，字串的輸入 `gets()` 函數，輸出調用 `puts()` 函數

```
Write something here : i love C language  
egaugnal C evol i  
請按任意鍵繼續 . . .
```

```
Write something here : Hello World  
dlrow olleH  
請按任意鍵繼續 . . .
```