Chapter 9
Array

Outline

- 1-D array
- 2-D array and multi-D array
- Passing arrays to functions
- Searching arrays
- Sorting arrays

1D Array

Arrays

- Arrays
  - Group of consecutive memory locations
  - Same name and type

Defining Arrays

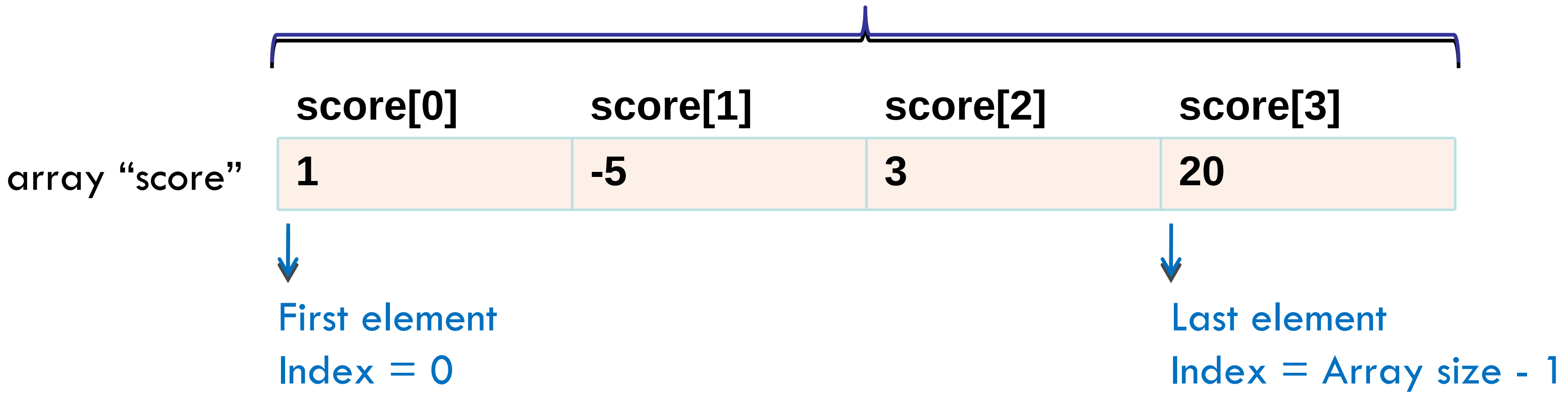
- When defining arrays, specify
  - Name
  - Type of array
  - Number of elements
- Format
  - Data_type array_name[number]
- Examples
  - int score[4]; /* integer arrays “score”, including 4 elements */
  - float temp[7]; /* float arrays “temp”, including 7 elements */
  - char name[6]; /* character arrays “name”, including 6 elements */

Defining Arrays (Cont.)

- Defining multiple arrays of same type
  - Format similar to regular variables
  - ***Example***
    - int arrayA[100], arrayB[27];
    - float arrayC[20], arrayD[25], arrayE[10];

Array Index

- Access an element of an array by index
- First element at position 0



- Array elements are like normal variables
  - score[2] = 3;
  - score[5-4] = -5;
  - printf("%d", score[3]);

Array Initialization

- Initializers
  - int n[5] = { 1, 2, 3, 4, 5 };
    - If not enough initializers, rightmost elements become 0
  - int n[5] = {0};
    - Set all elements to 0
  - int n[5] = {1} ;
    - Set n[0] = 1; n[1] ~ n[4] = 0
  - If too many initializers, a syntax error occurs
- If size omitted, initializers determine it
  - int n[ ] = { 1, 2, 3, 4, 5 };
  - 5 initializers, therefore 5 element array

Array Initialization (Cont.)

```
int main() {
    int score[5];
    score[0] = 90;
    score[1] = 80;
    score[2] = 75;
    score[3] = 88;
    score[4] = 65;
    for(int i = 0; i < 5; i++)
            printf("score[%d] = %d\n", i, score[i]);
    return 0;
}
```

Examples

```
int main() {
    int i, n[5];
    for(i = 0; i < 5; i++)
        n[i] = 2 * i;
    for(i = 0; i < 5; i++)
        printf("n[%d] = %d\n", i, n[i]);
    return 0;
}
```

output

```
n[0] = 0
n[1] = 2
n[2] = 4
n[3] = 6
n[4] = 8
```

Examples (Cont.)

```
int main() {
    int i, n[5] = {1};
    for(i = 0; i < 5; i++)
        printf(“n[%d] = %d\n”, i, n[i]);
    return 0;
}
```

output

```
n[0] = 1
n[1] = 0
n[2] = 0
n[3] = 0
n[4] = 0
```

Common Programming Error

```
int main() {
    int score[5];
    score[0] = 90;
    score[1] = 80;
    /* forget to initialize score[2] */
    score[3] = 88;
    score[4] = 65;
    for(int i = 0; i < 5; i++)
        printf("score[%d] = %d\n", i, score[i]);
    return 0;
}
```

ERROR: There is no value for score[2]

Another Example

```
/*  一維陣列的基本操作 */
#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;
}
```

```
/* OUTPUT---

score[0]=78
score[1]=55
score[2]=92
score[3]=80
--------------------*/
```

Another Example (Cont.)

```
/*  一維陣列的基本操作(錯誤的示範)  */
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   int i,score[4];

   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;
}
```

```
/* OUTPUT---

score[0]=78
score[1]=55
score[2]=51
score[3]=80
score[4]=2293600
----------------------*/
```

這兩個值都是原先留於記憶體內的殘值

sizeof()

- sizeof(array_name)
  - Return how many bytes the array occupies.

```
int main() {
    int data[5] = {0};
    printf("Size of array data (bytes): %d\n", sizeof(data));
    printf("Size of elements in data: %d\n", sizeof(data[0]));
    printf("Number of elements: %d\n", sizeof(data)/sizeof(data[0]));
    return 0;
}
```

output

```
Size of array data (bytes): 20
Size of elements in data: 4
Number of elements: 5
```

Read Data to an Array

```
int main() {
    int i, n[5];
    for(i = 0; i < 5; i++) {
           printf(“input element %d: ”, i);
           scanf(“%d”, &n[i]);
    }
    for(i = 0; i < 5; i++)
           printf(“element %d = %d\n”, i, n[i]);
    return 0;
}
```

Read Data to an Array (Cont.)

```c
/* 一維陣列內元素的設值 */
#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;
}
```

```
/* OUTPUT---

請輸入 age[0]的值:12
請輸入 age[1]的值:54
請輸入 age[2]的值:55
age[0]=12
age[1]=54
age[2]=55
-----------------------*/
```

Array Application - Maximal and minimal values

```
/* 比較陣列元素值的大小 */
#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;
}
```

```
/* OUTPUT---

陣列裡元素的最大值為 74
陣列裡元素的最小值為 17
----------------------*/
```

Another Way to Specify Array Size

Usually use uppercase letters

```
#define SIZE 10


int main() {
    int n[SIZE] = {0};
    return 0;
}
```

No semicolon is needed

Common Programming Error

- We can not set the variable as the number of elements.

```
int main() {
    int size = 10;
    int n[size] = {0};
    return 0;
}
```

Syntax error

Array Application
- Nondeterministic Number of Input Data

```
/* 輸入未定個數的資料到陣列裡 */
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
int main(void)
{
   int score[MAX];
   int i=0,num;
   int sum=0;
   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;
}
```

```
/* OUTPUT---

請輸入成績，要結束請輸入 0:
請輸入成績:70
請輸入成績:80
請輸入成績:60
請輸入成績:90
請輸入成績:0
平均成績為 75.00
-------------------------*/
```

Boundary Checking

```
01    /* 陣列的界限檢查 */
02    #include <stdio.h>
03    #include <stdlib.h>
04    #define MAX 5
05    int main(void)
06    {
07       int score[MAX];
08       int i=0,num;
09       float sum=0.0f;
10       printf("請輸入成績，要結束請輸入 0:\n");
11       do
12       {
13          if(i==MAX)  /* 當 i 的值為 MAX 時，表示陣列已滿，即停止輸入 */
14          {
15             printf("陣列空間已使用完畢!!\n");
16             i++;      /* 此行先將 i 值加 1，因為 23 行會把 i 的值減 1 掉 */
17             break;
18          }
19          printf("請輸入成績:");
20          scanf("%d",&score[i]);
21       }while(score[i++]>0);  /* 輸入 0 時結束 */
22       num=i-1;
23       for(i=0;i<num;i++)
24          sum+=score[i];         /* 計算平均成績 */
25       printf("平均成績為 %.2f\n",sum/num);
26
27       system("pause");
28       return 0;
29    }
```

```
/* OUTPUT---

請輸入成績，要結束請輸入 0:
請輸入成績:60
請輸入成績:50
請輸入成績:70
請輸入成績:80
請輸入成績:90
陣列空間已使用完畢!!
平均成績為 70.00
----------------------*/
```

C language never checks the boundary of arrays so as to enhance the execution performance.

Array Searching

```
/* 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;
}
```

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

陣列A 元素的值為:33 75 69 41 33 19
請輸入欲搜尋的整數:33
找到了! A[0]=33
找到了! A[4]=33
------------------------------*/
```

2D Array and Multi-D Array

2D Arrays

- Multiple subscripted arrays
  - Tables with rows and columns (m by n array)
  - Like matrices: specify row, then column
- Defining 2D arrays
  - **int** data[10][5]; /* 可存放10列5行個整數 */
  - **float** score[4][3]; /* 可存放4列3行個浮點數 */

Declaration of 2D Array

```
DataType ArrayName[RowNum][ColNum];
```

2D-Array Initialization

- Initializers grouped by row in braces
  - int b[2][3] = { { 1, 2, 3 }, { 4, 5, 6 } };
  - int b[2][3] = { { 1, 2, 3 },
    { 4, 5, 6 } };

1	2	3
4	5	6

- If not enough, unspecified elements set to zero
  - int b[2][3] = { { 1 }, { 4, 5 } };

1	0	0
4	5	0

2D-Array Initialization (Cont.)

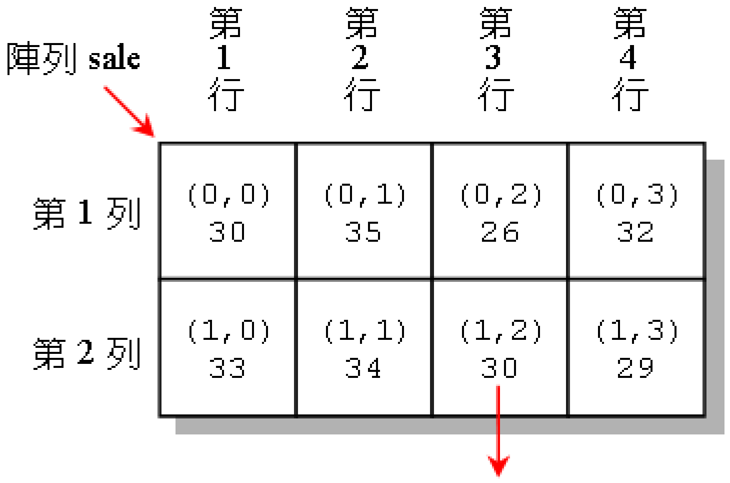
```
int main() {
    int i, j, b[5][5];
    /* set each element to 1*/
    for(i = 0; i < 5; i++)
            for(j = 0; j < 5; j++)
                    b[i][j] = 1;
    return 0;
}
```

Table and 2D Array

2D array is suitable for table handling.

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



	第 1 行	第 2 行	第 3 行	第 4 行
第 1 列	(0,0) 30	(0,1) 35	(0,2) 26	(0,3) 32
第 2 列	(1,0) 33	(1,1) 34	(1,2) 30	(1,3) 29

每一格代表一個元素，每個元素皆為 int 型態

Could be ignored

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}};
```

$\underbrace{\texttt{sale[2][4]}}_{\text{2×4 的陣列}}$ $\underbrace{\texttt{\{30,35,26,32\}}}_{\text{一維陣列，有 4 個元素}}$ $\underbrace{\texttt{\{33,34,30,29\}}}_{\text{一維陣列，有 4 個元素}}$

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

Easier to read

Accessing 2D Arrays

```
/*  二維陣列的輸入輸出 */
#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;
}
```

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

業務員 1 的第 1 季業績:30
業務員 1 的第 2 季業績:35
業務員 1 的第 3 季業績:26
業務員 1 的第 4 季業績:32
業務員 2 的第 1 季業績:33
業務員 2 的第 2 季業績:34
業務員 2 的第 3 季業績:30
業務員 2 的第 4 季業績:29
***Output***
業務員 1 的業績分別為 30  35  26  32
業務員 2 的業績分別為 33  34  30  29
2004 年總銷售量為 249 部車
------------------------------*/
```

Matrix Addition

$$A=\begin{bmatrix}1 & 2 & 3\\ 5 & 6 & 8\end{bmatrix};\quad B=\begin{bmatrix}3 & 0 & 2\\ 3 & 5 & 7\end{bmatrix}$$

$$A+B=\begin{bmatrix}1 & 2 & 3\\ 5 & 6 & 8\end{bmatrix}+\begin{bmatrix}3 & 0 & 2\\ 3 & 5 & 7\end{bmatrix}=\begin{bmatrix}1+3 & 2+0 & 3+2\\ 5+3 & 6+5 & 8+7\end{bmatrix}=\begin{bmatrix}4 & 2 & 5\\ 8 & 11 & 15\end{bmatrix}$$

```c
/* 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}};
   int B[ROW][COL]={{3,0,2},{3,5,7}};
   printf("Matrix A+B=\n");
   for(i=0;i<ROW;i++)      /* 外層迴圈 */
   {
      for(j=0;j<COL;j++)   /* 內層迴圈 */
         printf("%3d",A[i][j]+B[i][j]);  /* 計算二陣列相加 */
      printf("\n");
   }
   system("pause");
   return 0;
}
```

```
/* OUTPUT---

Matrix A+B=
  4  2  5
  8 11 15
-----------------------*/
```

Multiple Subscripted Arrays

- 3D array
  - Example
    - int array[2][3][4];
  - Initialization
    - int array[ ][3][4] = {

      {{1,2,3,4},
      {2,2,3,4},
      {3,2,3,4}},
      First 2D array

      {{4,2,3,4},
      {5,2,3,4},
      {6,2,3,4}}
      Second 2D array

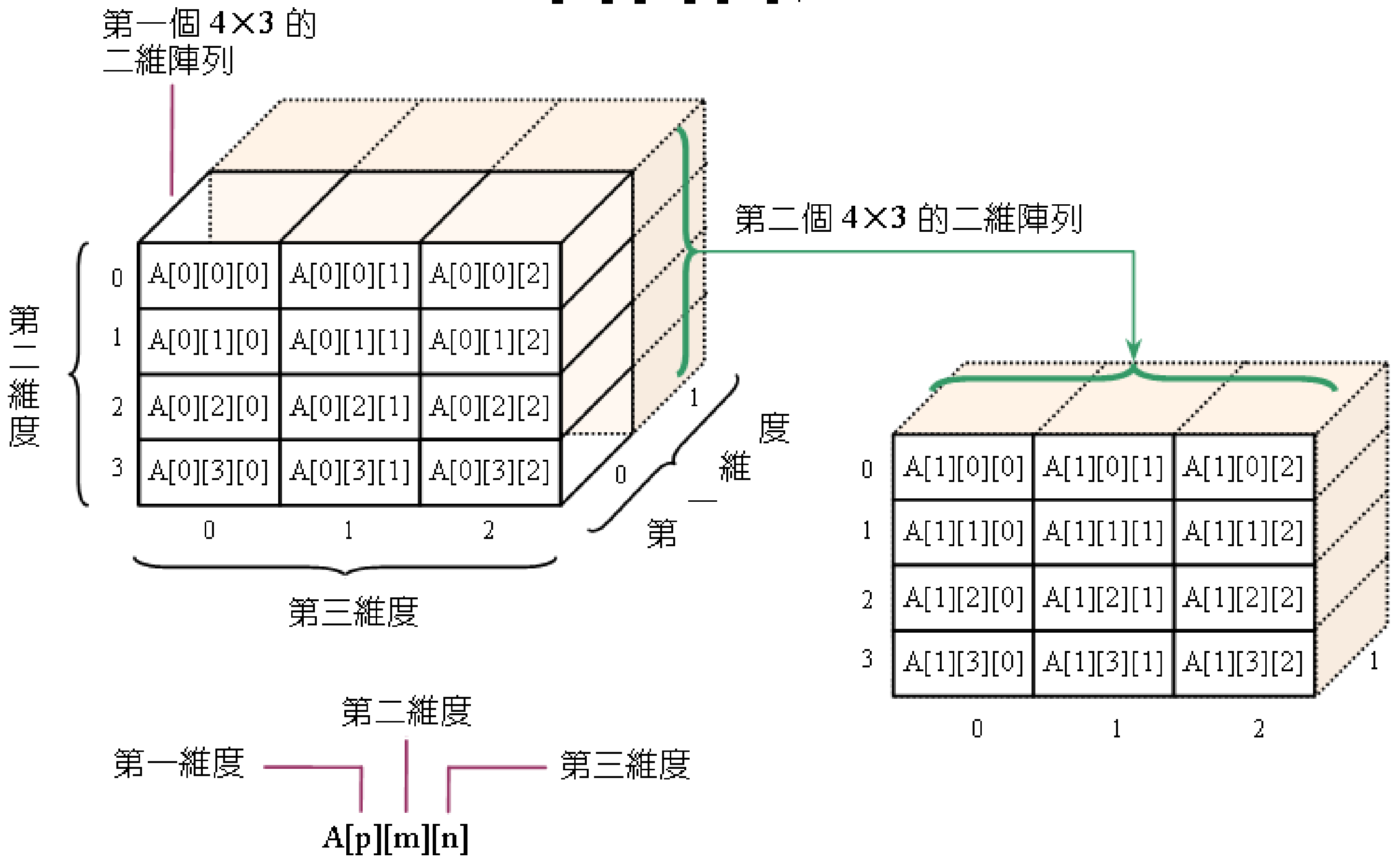
      };

      Could be ignored

    - for (i = 0; i < 2; i++)
        for (j = 0; j < 3; j++)
          for (k = 0; k < 4; k++)
            array[i][j][k] = 1;

Multiple Subscripted Arrays (Cont.)

```
int A[2][4][3];
```

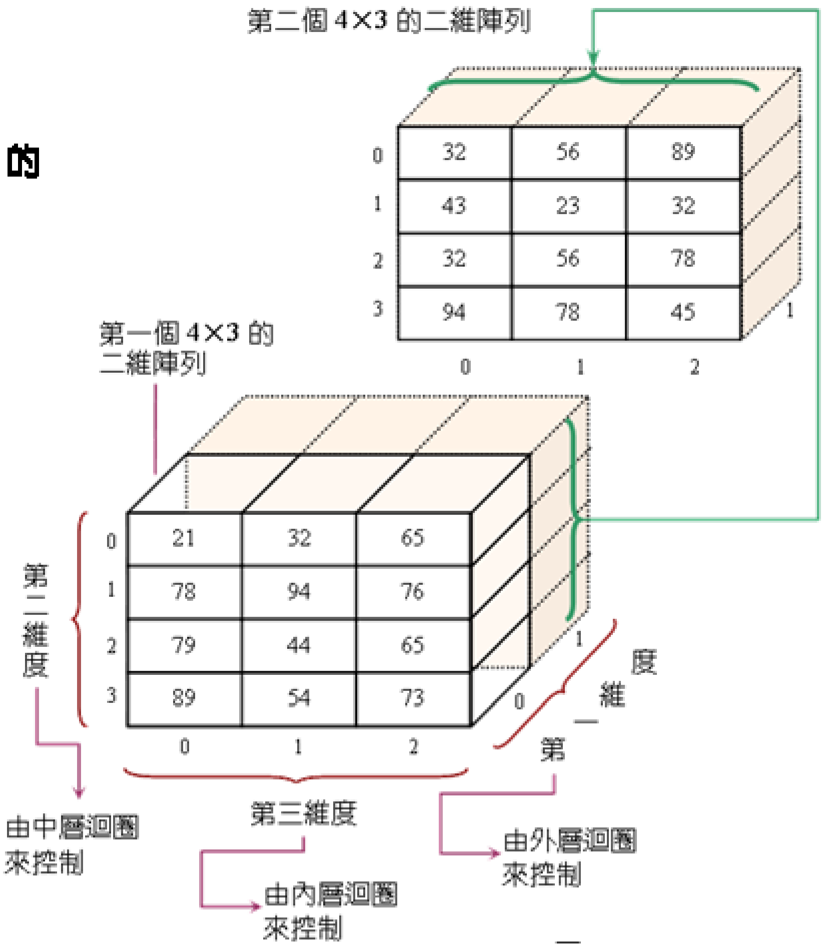


Finding the Maximal Value

```
/* prog9_11, 三維陣列與初值的設定 */
#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}}};
```

第一個 4×3 的二維陣列

第二個 4×3 的二維陣列



Finding the Maximal Value (Cont.)

```
    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;
}
```

利用三個 for 迴圈找出陣列的最大值

```
/* OUTPUT---
max=94
---------------------*/
```

Passing Arrays to Functions

Passing Arrays to Functions

Passing 1D Array

```
ReturnType FuncName(DataType Arrayname[]);  /* Declaration */
int main(void)
{
   DataType ArrayName[NumOfElements];
       ...
   FuncName(ArrayName);
     ...
}
ReturnType FuncName(DataType ArrayName[] )
{
    ...
}
```

Size of Array could be ignored

Passing Arrays to Functions (Cont.)

- Parameter names optional in prototype
  - int b[] could be written int []
  - int arraySize could be simply int
- Arrays passed call-by-reference
- Name of array is address of its first element

Example

```
/* OUTPUT---
陣列的內容為: 5 3 6 1
------------------------*/
```

```
/* 傳遞一維陣列到函數裡 */
#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");
}
```

Array Address

- The address of the first element is the array’s address.

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

```
/* OUTPUT----

A[0]=20,位址=0022FF48
A[1]= 8,位址=0022FF4C
A[2]=13,位址=0022FF50
陣列 A 的位址=0022FF48
---------------------------*/
```

Call by Value

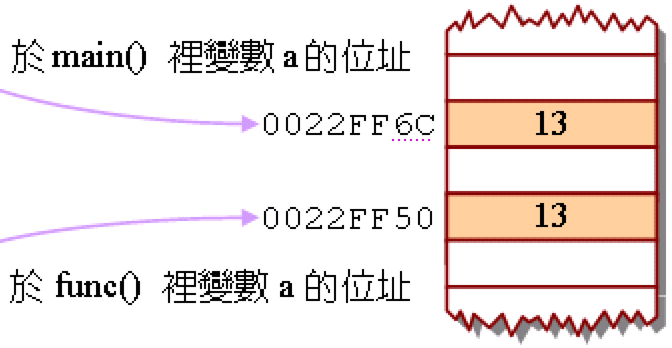
```
/* 印出變數的位址 */
#include <stdio.h>
#include <stdlib.h>
void func(int);
int main(void)
{
   int a=13;
   printf("於main()裡,a=%d,a的位址=%p\n",a,&a);
   func(a);              /* 這是傳值呼叫的機制 */

   system("pause");
   return 0;
}

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

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

於main()裡,a=13,a的位址=0022FF6C
於func()裡,a=13,a的位址=0022FF50
------------------------------------*/
```



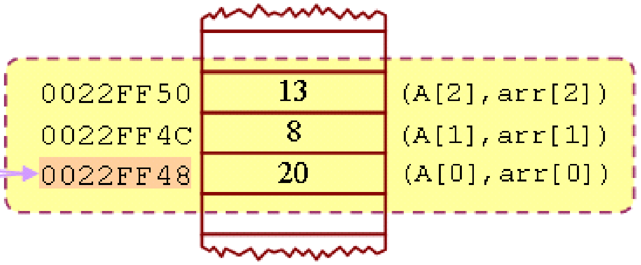
Call by Address

```
/* 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[])
{
   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]);
}
```

```
/* prog9_14 OUTPUT--------

在 main()裡，陣列 A 元素的位址為
A[0]=20,位址為 0022FF48
A[1]= 8,位址為 0022FF4C
A[2]=13,位址為 0022FF50

在 func()裡，陣列 arr 元素的位址為
arr[0]=20,位址為 0022FF48
arr[1]= 8,位址為 0022FF4C
arr[2]=13,位址為 0022FF50
---------------------------*/
```



Application of Call by Address

```
/* 於函數內更改陣列元素的值 */
#include <stdio.h>
#include <stdlib.h>
#define SIZE 4
void show(int arr[]);
void add2(int arr[]);

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;
}
```

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

呼叫 add()前,陣列的內容為: 5 3 6 1
呼叫 add()後,陣列的內容為: 7 5 8 3
-------------------------------*/
```

Passing 2D Arrays

Declaration of 2D Array

```
ReturnType FuncName(DataType ArrayName[][ElementNum]);
int main(void)
{
   DataType ArrayName[RowNum][ColNum];
    ...
   FuncName(ArrayName);
    ...
}
ReturnType FuncName(DataType ArrayName[][ColNum] )
{
    ...
}
```

Must fill in

Must fill in

Example
– Finding the Maximal/Minimal Value

```
/* 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");
   for(i=0;i<ROW;i++)
   {
      for(j=0;j<COL;j++)
         printf("%02d ",a[i][j]);
      printf("\n");
   }
```

Example – Finding the Maximal/Minimal Value (Cont.)

```
  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])    /* 尋找最大值 */
        p[0]=arr[i][j];
      if(p[1]>arr[i][j])    /* 尋找最小值 */
        p[1]=arr[i][j];
    }
}
```

```
/* OUTPUT---

二維陣列內的元素:
26 05 07
10 03 47
06 76 08
40 04 32
陣列的最大值=76
陣列的最小值=03
------------------------*/
```

Searching and Sorting Arrays

Searching Arrays

```
#include <stdio.h>
#define SIZE 5
int search(int array[], int size, int key) {
      for (int i = 0; i < size; i++)
            if(array[i] == key)
                  return i;
      return -1;
}
int main() {
      int array[SIZE] = {20, 25, 30, 35, 40};
      int search1 = search(array, SIZE, 35);
      int search2 = search(array, SIZE, 44);
      printf("element of (35) = %d\nelement of (44) = %d\n", search1, search2);
      return 0;
}
```

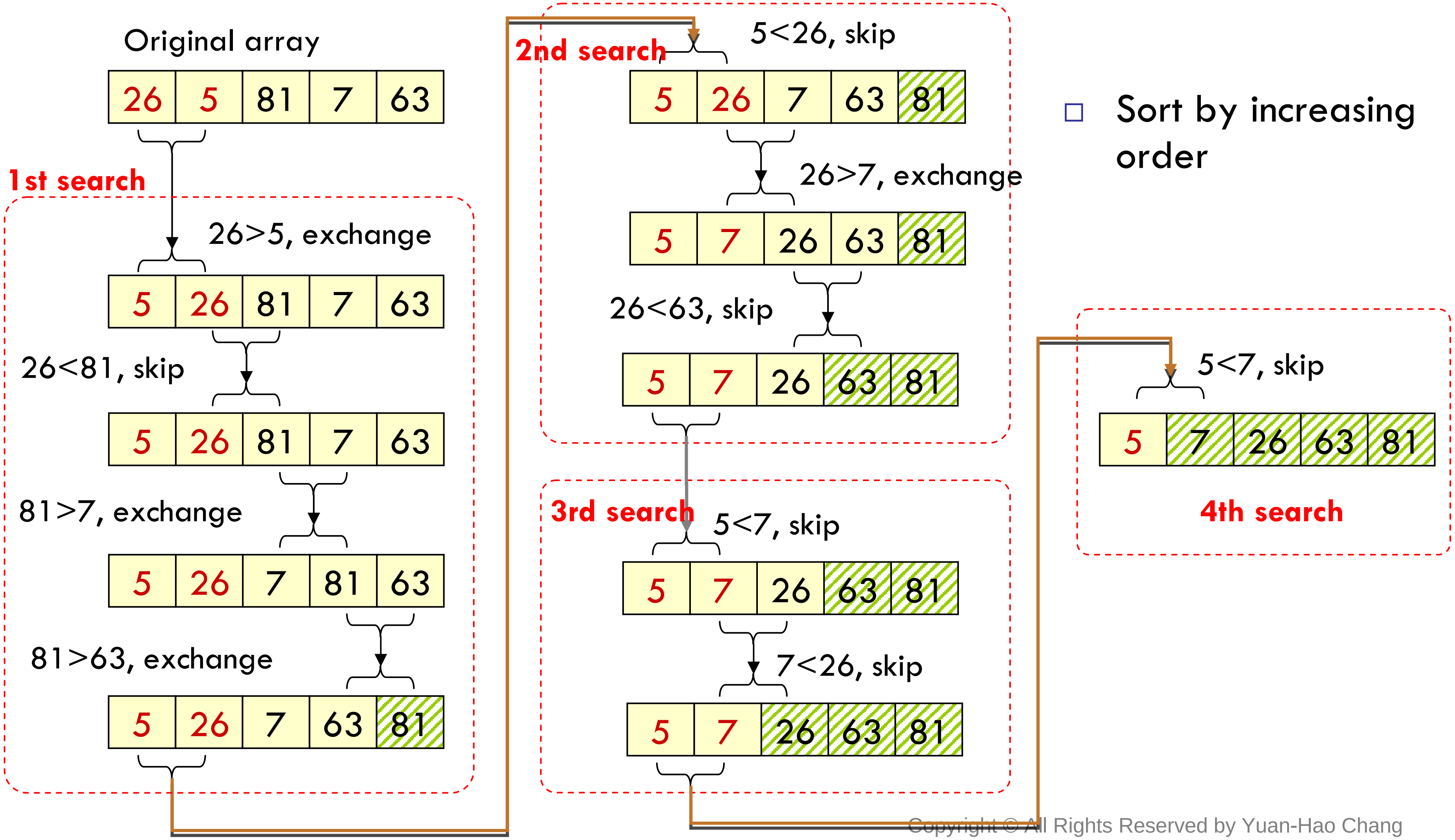
output

```
element of (35) = 3
element of (44) = -1
```

Sorting Arrays

- Bubble sort (sinking sort)
  - Several passes through the array
  - Successive pairs of elements are compared
    - If increasing order (or identical), no change
    - If decreasing order, elements exchanged
  - Repeat

Sorting Arrays – Bubble Sort



- Sort by increasing order

Bubble Sort

```c
/* 氣泡排序法 */
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
void show(int a[]), bubble(int a[]);
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");
}
```

```
/* OUTPUT---
排序前...
26 5 81 7 63
排序後...
5 7 26 63 81
--------------------------*/
```

Bubble Sort (Cont.)

```
25  void bubble(int a[])
26  {
27     int i,j,temp;
28     for(i=1;i<SIZE;i++)
29        for(j=0;j<(SIZE-i);j++)
30           if(a[j]>a[j+1])
31           {
32              temp=a[j];
33              a[j]=a[j+1];
34              a[j+1]=temp;
35           }
36  }
```

如果 a[j]>a[j+1]，則元素的值互換

26	5	81	7	63

原始陣列

第一次搜尋，i=1，j=0~3

	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=0~2

	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=0~1

	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

Lab 09-1

- Write a program to declare an array with 5 elements. Then use for loop to assign arr[0]~arr[4] to 1~5, respectively. Finally print out the value in each array element.
- Declare an array int array = {1, 2, 3, 4, 5, 6}. Use sizeof() to calculate and output the number of elements in this array, the size (i.e., the number of bytes) of this array.
- Write a program to calculate the result of multiplying the following two matrices. $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$
- Write a function ***double average(int arr1[][2], int arr2[][2])*** to return the average of the 8 elements in arr1[][] and arr2[][], where the two arrays are listed in the above.

Lab 09-2

- Write a program to answer the following questions:
  - The sale amount of each salesman.
  - The sale amount of each product.
  - Who is the best salesman?
  - Which product has the higher sale amount.

Sales	A	B	C	D	E
1	3	2	6	5	3
2	7	3	8	5	3
3	3	5	3	7	5
Price	5	4	6	7	3

- Write a program to answer the following questions:
  - Print out the content of arrays.
  - Average temperature of each day.
  - Average temperature of each time slot.
  - The time slot and day with the highest temperature.
  - The time slot and day with the lowest temperature.

	Mon	Tue	Wed	Thu
T1	18.2	17.3	15.0	13.4
T2	23.8	25.1	20.6	17.8
T3	20.6	21.5	18.4	15.7