

# CH7 陣列與向量 Array and Vectors

課程名稱：資管一程式設計  
任課教師：謝明哲  
單位職稱：台東大學資管系副教授  
電子郵件：hmz@nttu.edu.tw

hmz@nttu.edu.tw 2016

1

## 什麼是陣列？

hmz@nttu.edu.tw 2016

3

## Outline

- 什麼是陣列？
- 陣列的運用
- 排序方式
- 多維陣列

hmz@nttu.edu.tw 2016

2

- 陣列(array)是一群具有相同型態的元素集合起來的特殊型態，每一個元素都有一個索引值(index)作為存取的依據。宣告時，在陣列名稱後加上中括號 [ ]，中括號內可以寫上此陣列的大小。

- 宣告方式：

資料型別 陣列名稱 [長度];

或

資料型別 陣列名稱 [長度] = {初始值0, 初始值1, ..., 初始值n-1};

可略

hmz@nttu.edu.tw 2016

4

■ ex.

```
int a[10];    //宣告陣列a為10個整數空間  
float f[20]; //宣告陣列f為20個浮點數空間  
char str[40]; //宣告陣列str為40個字元空間
```

**注意：**因為陣列是連續的儲存單元，所以每個單元都是相同的型態！！

字元陣列的宣告及應用：  
輸入6個字元，以相反順序列印出來

```
char str[6];  
cout<< "input 6 characters string: " ;  
for(int i=0;i<6;i++) cin>>str[i];  
cout<<endl;  
for(int i=5;i>=0;i--) cout<<str[i];
```

## 陣列的運用

整數陣列的宣告初值設定及使用：  
求出5人中的最高與最低成績

```
int score[5] = {87,88,91,76,99};  
int max,min;  
max=min=score[0];  
for(int i=1;i<5;i++){  
    if (score[i]>max) max=score[i];  
    if (score[i]<min) min=score[i];  
}
```

### 模擬丟骰子6000次並統計各點出現次數

```
int freq[7]={0};
for(int i=0;i<6000;i++)
{
    int die = rand()%6+1;
    freq[die]++;
}
for (int i=1;i<=6;i++)
    cout<<i<<" : " <<freq[i]<<endl;
```

[0]	[1]	[2]	[3]	[4]	[5]	[6]
-----	-----	-----	-----	-----	-----	-----

如果擲出骰子的骰子數為1，則freq[1]+1  
其它可依此類推

### 兩種常用的排序演算法：

1. 選擇排序 Selection sort
2. 泡沫排序 Bubble sort

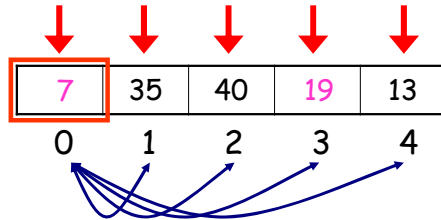
## 排序方式

### Selection sort

- 步驟 1：一開始整個數列歸類為未排序。
- 步驟 2：從未排序的數列中，挑選出最小的數，並與第一個元素的位置互換，並將此最小的數，歸類為已排序數列。
- 步驟 3：重複步驟 2，直到所有的數都歸到已排列數列中。

N = 5

Run 1:



hmz@nttu.edu.tw 2016

13

```
for (int i=0; i<n; i++)  
{  
    for (int j=i+1; j<n; j++)  
        if (A[i]>A[j]) swap(A[i], A[j]);  
}
```

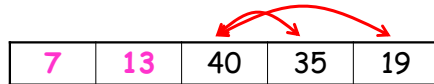
hmz@nttu.edu.tw 2016

15

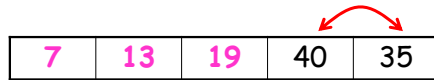
Run 2:



Run 3:



Run 4:



hmz@nttu.edu.tw 2016

14

## Bubble sort

**步驟 1**：一開始整個數列歸類為未排序。

**步驟 2**：從未排序的數列中的第一個數開始看，如果前面的數比後面的數，就往後推。在這過程中，最大的數會被推到未排列數列中的最後一個位置，將該最大的數歸類到已排序的數列中。

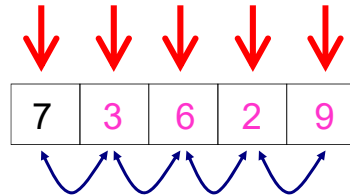
**步驟 3**：重複步驟 2，直到沒有往後推的動作為止。

hmz@nttu.edu.tw 2016

16

N = 5

Run 1:



hmz@nttu.edu.tw 2016

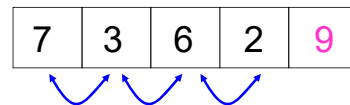
17

```
void bubbleSort (int x[ ], int n)
{
    for (int i=0;i<n-1;i++)
        for (int j=0;j<n-1-i;j++)
            if (x[j+1] < x[j])
                swap(x[j], x[j+1]);
}
```

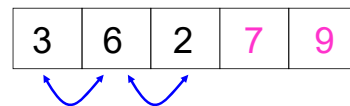
hmz@nttu.edu.tw 2016

19

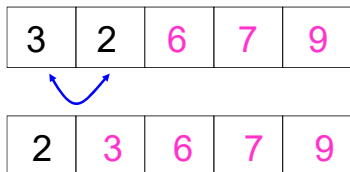
Run 2:



Run 3:



Run 4:



hmz@nttu.edu.tw 2016

18

## 多維陣列

hmz@nttu.edu.tw 2016

20

語法：

型別 陣列名[n][m];

或

型別 陣列名[n][m]={{...},{...},...,{...}};

範例：

//宣告一個2x4整數二維陣列

int a[2][4];

//宣告一個2x4整數二維陣列，並設定初始值

int a[2][4] = { {0,1,2,3}, {4,5,6,7} };

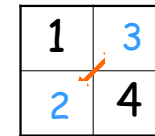
hmz@nttu.edu.tw 2016

21

## 轉換矩陣

```
void transpose(int A[][2], int n)
{
    for (int i=0;i<n;i++)
        for (int j=i+1;j<n;j++)
            swap(A[i][j], A[j][i])
}

void main()
{
    int x[][2] = { {1, 2}, {3, 4} };
    transpose(x, 2);
}
```



1	3
2	4

hmz@nttu.edu.tw 2016

23

$$a[2][4] = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

a[0][0] a[0][1] a[0][2] a[0][3]  
a[1][0] a[1][1] a[1][2] a[1][3]

hmz@nttu.edu.tw 2016

22

## 使用指標傳遞矩陣 (傳址呼叫法)

```
void transpose(int *ptrA, int n)
{
    for (int i=0;i<n;i++)
        for (int j=i+1;j<n;j++)
            swap(*(ptrA+i*n+j), *(ptrA+j*n+i));
}

void main()
{
    int X[][3] = { {0, 1, 2},
                  {3, 4, 5},
                  {6, 7, 8},
                  {9, 10, 11} };
    int *ptrX = &X[0][0];
    transpose(ptrX, 3);
}
```

記憶體位址	X[][3]內容
ptrA+0	0
ptrA+1	1
ptrA+2	2
ptrA+3	3
ptrA+4	4
ptrA+5	5
ptrA+6	6
ptrA+7	7
ptrA+8	8
ptrA+9	9
ptrA+10	10
ptrA+11	11

hmz@nttu.edu.tw 2016

24

# 標準樣版函式庫STL

## Standard Template Library

hmz@nttu.edu.tw 2016

25

### ■ vector物件的運算

#### □ 設定

```
x = y;
```

#### □ 相等

```
if (x==y) cout<<" x ==y" ;
```

#### □ 長度

```
for(int i=0;i<x.size();i++) cin>>x[i];
```

#### □ 加入元素

```
x.push_back(0); //在x末尾添加元素0
```

hmz@nttu.edu.tw 2016

27

### STL vector物件 (template類別)

#### ■ #include <vector>

#### ■ using std::vector;

#### ■ 使用vector類別宣告一維陣列(vector物件)

```
vector<int> x(5); // 5個整數未設定初始值
```

```
vector<int> y(8,0); // 8個整數初始值為0
```

```
vector<int> z(y); // 以y初始z，z為y的複製
```

hmz@nttu.edu.tw 2016

26

#### □ 讀取末尾元素

```
cout<<x.back(); //列印x末尾元素
```

#### □ 讀取指定元素

```
cout<<x.at(0); //列印x第0個元素
```

```
cout<<x[0]; //列印x第0個元素
```

#### □ 刪除末尾元素

```
x.pop_back(); //刪除x末尾元素
```

#### □ 清除所有元素

```
x.clear(); //將x清空
```

hmz@nttu.edu.tw 2016

28

## bubbleSort()的vector版

```
void bubbleSort (vector<int> &x)
{
    for (int i=0; i<n-1; i++)
        for (int j=0; j<x.size()-1-i; j++)
            if (x[j+1]<x[j]) swap(x[j], x[j+1]);
}
```

hmz@nttu.edu.tw 2016

29

## STL map物件 (template類別)

- #include <map>
- using std::map;
- 使用map類別宣告<座號,姓名>映射物件  
`map<int, string> student;`
- 加入<座號,姓名>映射值
  1. `student.insert(pair<int, string>(1, "李誠" ));`
  2. `student.insert(  
 map<int, string>::value_type (1, "李誠" ));`
  3. `student[1] = "李誠" ;`

hmz@nttu.edu.tw 2016

31

## STL的sort()排序演算法

- #include <algorithm>
- 宣告Vector物件
  - `vector<Grade> grade;`
- 自行定義 < 的operator函式給sort()函式使用
  - `bool operator<(const Grade &x, const Grade &y)`

```
{
    if (x.average < y.average) return true;
    else return false;
}
```
- 使用sort()函式由小到大進行排序
  - `sort(&grade[0], &grade[N]); //N=grade.size()`

hmz@nttu.edu.tw 2016

30

- 取得map的大小
  - `cout<<student.size();`
- 由key尋找映射值
  - `map<int, string>::iterator iter;`
  - `iter = student.find(1);`
  - `if (iter != student.end())`

```
    cout << iter->first << " , " << iter->second;
```

`else cout<<" not find" ;`
- 刪除映射值
  - `student.erase(iter);`
- 清除所有映射值
  - `student.clear();`

hmz@nttu.edu.tw 2016

32



## STL set物件 (template類別)

- #include <set>
- using std::set;
- 使用set類別宣告集合物件  
`set<int> s;`
- 加入集合元素，不可重複
  - `for (int i=0;i<N;i++) s.insert(i);`
- 列印所有集合元素
  - `set<int>::iterator iter;`
  - `for (iter=s.begin();iter!=s.end();iter++)`  
`cout<<*iter<<" ";`

hmz@nttu.edu.tw 2016

33

## 樂活時間

- 為何哈佛飲食金字塔建議每天適量食用「健康油脂」，避免「不健康油脂」？
- 為何哈佛飲食金字塔建議少吃紅肉&奶油？

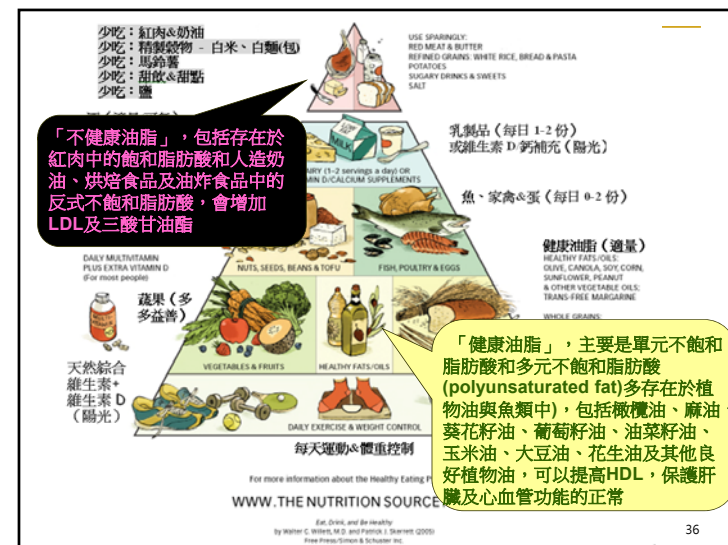
hmz@nttu.edu.tw 2016

35

- 加入集合元素，但不可重複
  - `s.insert(3);`
- 尋找集合元素 '3'
  - `iter=s.find(3);`
  - `if (iter!=s.end()) cout<<*iter;`
- 刪除集合元素 '3'
  - `s.erase(3);`
- 判斷是否為空集合
  - `if (s.empty()) cout<<"空集合";`

hmz@nttu.edu.tw 2016

34



36

## 習題

### Ch7. Exercises

7.8 (6%) 7.9 (14%)

7.11(6%) 7.13(8%) 7.14(8%) 7.16(6%) 7.17(6%) 7.18(6%)

7.21 (6%)

7.31 (6%) 7.35 (6%) 7.36 (6%) 7.37 (6%)