第六章

陣列與字串

前言

- 陣列是一種非常重要的資料結構，它可以讓程式設計更精簡，甚至還可以配合硬體的設計提昇程式的效率。

- 字串可以視爲一種資料型態，但C語言實作字串時，則將字串實作爲一種特殊的字元陣列。

大綱

- 6.1 陣列
  - 6.1.1 宣告陣列
  - 6.1.2 一維陣列
  - 6.1.3 氣泡排序法 (Bubble Sort)【補充】
  - 6.1.4 二維陣列
- 6.2 字串
  - 6.2.1 字元陣列與字串的宣告：
  - 6.2.2 字串陣列的宣告
  - 6.2.3 字串整合範例

大綱

- 6.3 字串相關函式
  - 6.3.1 輸入與輸出－gets( )、fgets( )與puts( )
  - 6.3.2 計算字串長度－strlen( )
  - 6.3.3 複製字串－strcpy( )與strncpy( )
  - 6.3.4 字串連結－strcat( )與strncat( )
  - 6.3.5 字串比較－strcmp( )與strncmp( )
  - 6.3.6 句元分割－strtok( )
- 6.4 本章回顧

6.1 陣列

- 什麼是陣列呢？簡單來說，陣列是一種儲存大量同性質資料的良好環境。
  - 『陣列』與數學的「矩陣」非常類似。也就是每一個陣列中的元素都有它的編號。
  - 『陣列』是一群資料型態相同的變數，並且在記憶體中會以連續空間來加以存放。
- 陣列中每個元件（陣列元素）相當於一個變數。

6.1 陣列

- 我們只要透過索引(index)，就可以直接取得陣列的指定元素（C語言的陣列索引由0開始計算）。
  - 例如：我們使用Month[0]~Month[11]來存放12個月份的營業額，當我們希望取出8月份的營業額時，則只要使用Month[7]當做變數名稱即可輕鬆取出該元素值。因此，使用陣列可以免除大量變數命名的問題，使得程式具有較高的可讀性。
- 事實上，在程式執行時，使用陣列有的時候還會加快運算速度，這與資料存放在連續記憶體有關，俗稱資料的區域性。

6.1.1 宣告陣列

- 陣列依照編號排列方式、佔用的空間大小，可以分爲一維陣列、二維陣列...等等，而在C語言中，陣列的使用與變數一樣，必須先經由宣告才可以使用，存取陣列的元素資料時，則採用索引值來指定要存取的陣列元素。

- 陣列的宣告必須指定陣列元素的資料型態，當陣列宣告完畢，我們就可以存取陣列中的元素資料了，以下是陣列宣告語法：

語法：資料型態 陣列名稱[第1維度元素個數] [第2維度元素個數]‥‥;
功能：宣告一維（二維‥‥）陣列，以及陣列元素的資料型態。

6.1.1 宣告陣列

- 【範例】：假設我們有12個月的營業額要記錄、而營業額爲整數，此時您可以使用Month也可以使用Trade（交易）做爲陣列名稱，如下宣告1維陣列：
  - int Trade[12];
  - ↑
  - 月份
  - 則Trade[7]代表8月份的營業額。（因爲C語言的陣列從索引0開始計算）

  - 【範例】：假設我們有兩年的每月營業額要記錄，則可以如下宣告2維陣列：
    - int Trade [2][12];
    - ↑ ↑
    - 年 月份
  - 則Trade[1][7]代表第2年8月份的營業額。

6.1.2 一維陣列

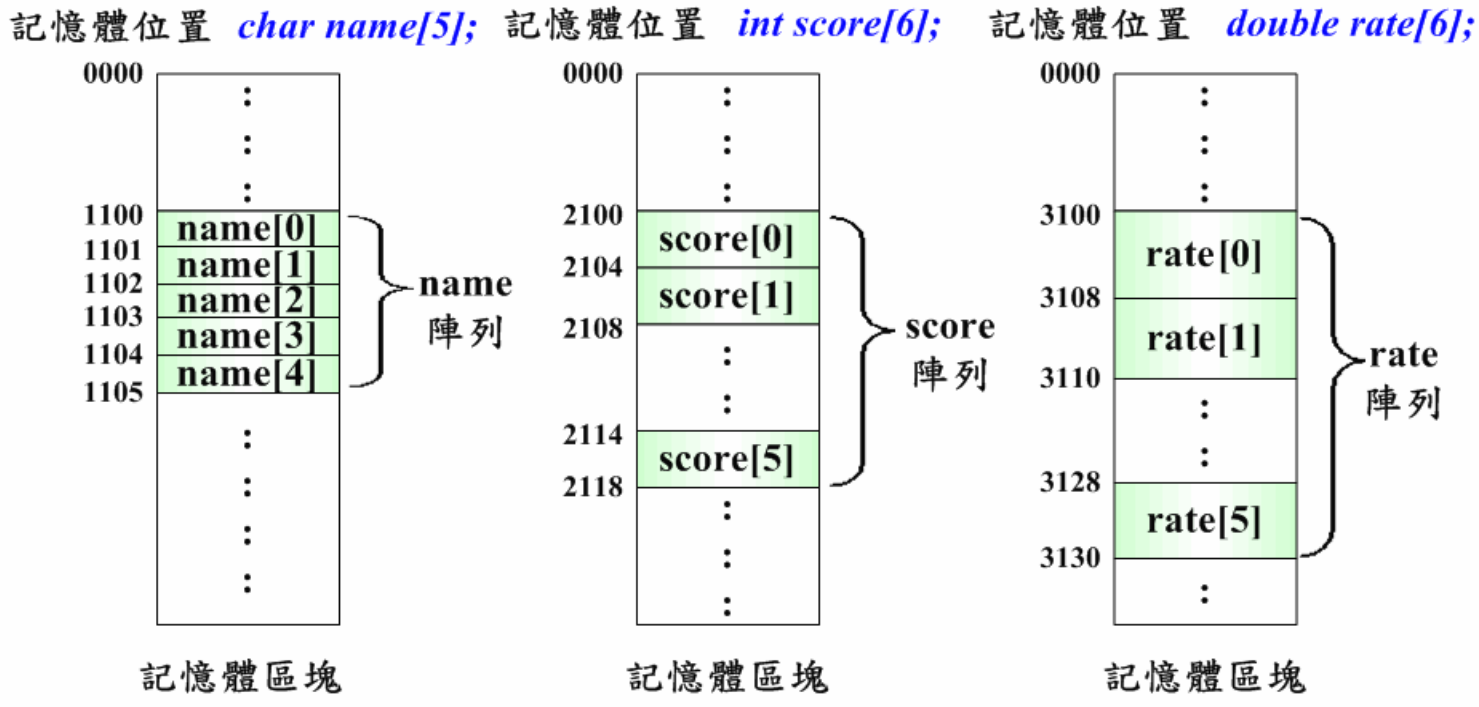
- 一維陣列的宣告敘述如下：

語法：資料型態　　陣列名稱[元素個數];
功能：宣告一維陣列，以及陣列元素的資料型態。

- 【範例】：
  - char　name[5]；　　/* 長度爲5的字元陣列 */
  - int　score[6]；　　/* 長度爲6的整數陣列 */
  - double　rate[6]；　　/* 長度爲6的浮點數陣列 */

- 【語法說明】：
  - 1. 陣列經過宣告後，系統將保留一塊連續的記憶體空間來存放陣列元素，記憶體內容如下範例。

6.1.2 一維陣列



陣列的記憶體配置圖

由上圖中，我們可以很容易地發現，陣列的各個元素依照次序存放在連續的記憶體中，這就是陣列的特性。

6.1.2 一維陣列

- score與rate陣列雖然元素個數（或稱爲陣列長度）都是6，但由於資料型態佔用的記憶體大小不同，因此兩個陣列所佔用的總記憶體空間也不相同。在後面的範例中，我們將透過sizeof( )運算子來求出陣列所佔記憶體空間的大小。
- 陣列一經宣告，就可以透過索引存取陣列元素，語法如下。在C語言中，陣列第一個元素的索引值爲0，第二個元素的索引值爲1，依此類推，所以長度爲n的陣列，其索引值爲0～n-1。

語法：陣列名稱[索引值]
功能：存取陣列元素。

6.1.2 一維陣列

- 範例6-1：ch6_01.c

```
1  /*****************************
2      檔名:ch6_01.c
3      功能:計算陣列大小
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   char name[5];
10  int score[6];
11  double  rate[6];
12   printf("name[5]共使用%dbytes\n",sizeof(name));
13  printf("score[6]共使用%dbytes\n",sizeof(score));
14  printf("rate[6]共使用%dbytes\n",sizeof(rate));
15  /*  system("pause");  */
16 }
```

- 執行結果：

name[5]共使用5bytes
score[6]共使用24bytes
rate[6]共使用48bytes

6.1.2 一維陣列

- 【實用範例6-2】：使用陣列存放氣溫資料，並計算平均溫度。
- 範例6-2：ch6_02.c

```
1  /*************************
2      檔名:ch6_02.c
3      功能:陣列元素的存取
4   *************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7   void main(void)
8  {
9    float Temper[12],sum=0,average;
10   int i;
11   for(i=0;i<12;i++)
12   {
13      printf("%d月的平均溫度:",i+1);
14      scanf("%f",&Temper[i]);
15      sum=sum+Temper[i];
16   }
17   average=sum/12;
18   printf("=======================\n");
19   printf("年度平均溫度:%f\n",average);
20   /*  system("pause");  */
21 }
```

6.1.2 一維陣列

- 執行結果：

```
1月的平均溫度:15.6
2月的平均溫度:17.3
3月的平均溫度:24.2
4月的平均溫度:26.7
5月的平均溫度:28.4
6月的平均溫度:30.2
7月的平均溫度:29.6
8月的平均溫度:30.5
9月的平均溫度:29.2
10月的平均溫度:28.6
11月的平均溫度:25.4
12月的平均溫度:22.9
==========
年度平均溫度:25.716667
```

- 範例說明：
  - 使用迴圈將輸入的溫度逐一存入Temper陣列的Temper[i]元素中，並且累加溫度總和。最後計算平均溫度。

6.1.2 一維陣列

- 一維陣列初始化
  - 我們在宣告陣列的同時，也可以指定陣列元素的初始值，語法如下：

語法：資料型態 陣列名稱[元素個數]={元素1初始值,元素2初始值,...};
功能：宣告一維陣列並設定陣列元素的初始值。

  - 【語法說明】：
    - 1. 陣列的元素個數不一定要明確指定，因爲編譯器可以根據初始值的個數，自行判斷該陣列的元素個數。
    - 【範例】：

```
float
Temper[12]={15.6,17.3,24.2,26.7,28.4,30.2,29.6,30.5,29.2,28.6,25.4,22.9};
                                                    相當於
float Temper[]={15.6,17.3,24.2,26.7,28.4,30.2,29.6,30.5,29.2,28.6,25.4,22.9};
```

6.1.2 一維陣列

- 2. 當明確定義陣列大小時，若設定的陣列元素初始值不足，則編譯器會將剩餘未設定初值的元素內容設定爲0（針對數值陣列而言）。
  - 【範例】：

    float Temper[12]={0}; /* 同時將12個元素內容皆設定爲0 */。

- 【觀念範例6-3】：陣列元素初始化與未初始化的比較。
- 範例6-3：ch6_03.c（檔案位於隨書光碟ch06\ch6_03.c）。

6.1.2 一維陣列

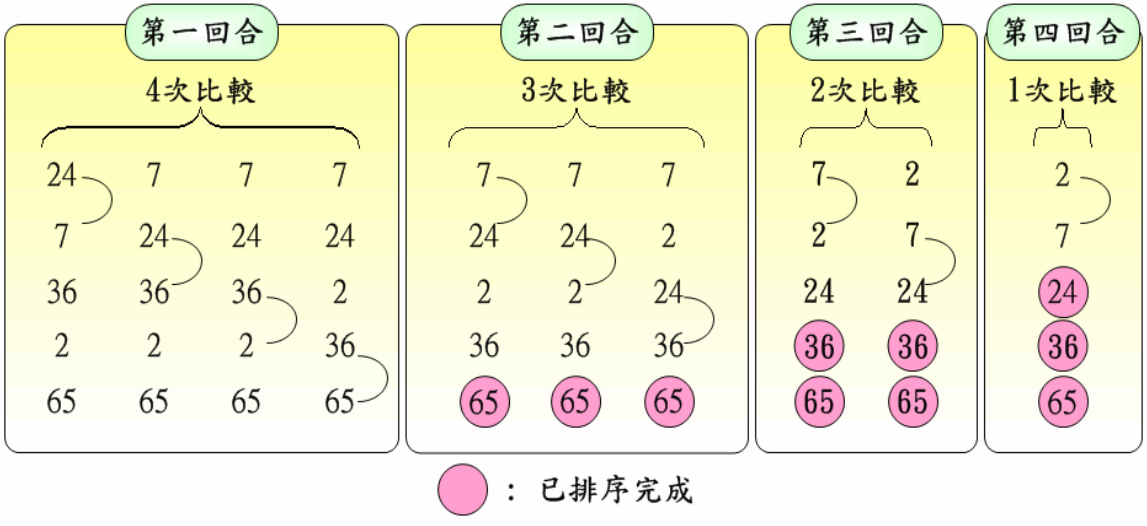
```
1  /*************************
2      檔名:ch6_03.c
3      功能:陣列元素初始化
4   *************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   float data1[10];
10  float data2[10]={0};
11  int i;
12   for(i=0;i<10;i++)
13  {
14     printf("data1[%d]=%.2f\n\t\t\t
15 data2[%d]=%.2f\n",i,data1[i],i,data2[i]);
16  }
17  /*  system("pause");  */
    }
```

6.1.3氣泡排序法 (Bubble Sort)【補充】

- 搜尋與排序是程式設計的一項基本且重要的問題。
  - 『搜尋』（Searching），指的是在一堆資料中，尋找您所想要的資料，例如：在英文字典中找尋某一個單字。
  - 『排序』（Sorting）則是將一堆雜亂的資料，依照某個鍵值（Key Value）依序排列，方便日後的查詢或使用。例如：英文字典中每個單字就是已經排序後的結果『從a~z』。

6.1.3氣泡排序法 (Bubble Sort)【補充】

- 『氣泡排序法』是一種非常簡單且容易的排序方法，簡單地來說，『氣泡排序法』是將相鄰兩個資料一一互相比較，依據比較結果，決定資料是否需要對調，由於整個執行過程，有如氣泡逐漸浮上水面，因而得名。
- 假設我們有{24,7,36,2,65}要做氣泡排序，最後的排序結果爲{2,7,24,36,65}。



氣泡排序法圖

6.1.3氣泡排序法 (Bubble Sort)【補充】

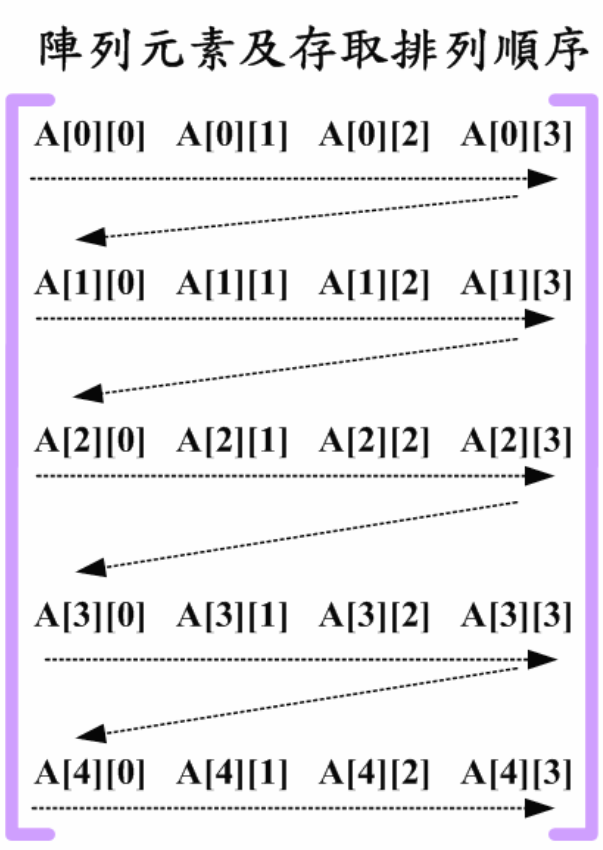
```
1   #include <stdio.h>
2   void main()
3   {
4       int i=0,j=0,temp=0,n=0,k=0,key=0;
5       int Data[10]={10,78,45,89,01,05,23,54,20,32};
6       for(i=0;i<10;i++)
7               cout <<Data[i]<<"\t";
8       cout <<"\n"<<"//////////////////////"<<"\n";
9       n=i;
10      for(i=0;i<10;i++)
11      {
12              key=0;
13              for(j=0;j<n-i-1;j++)
14              {
15                      if(Data[j]<Data[j+1])
16                      {
17                              temp=Data[j];Data[j]=Data[j+1];Data[j+1]=temp;key=1;
18                      }
19              }
20              if(key==0)
21                      break;
22              for(k=0;k<10;k++)
                        cout <<Data[k]<<"\t";
                cout <<"\n";
        }
}
```

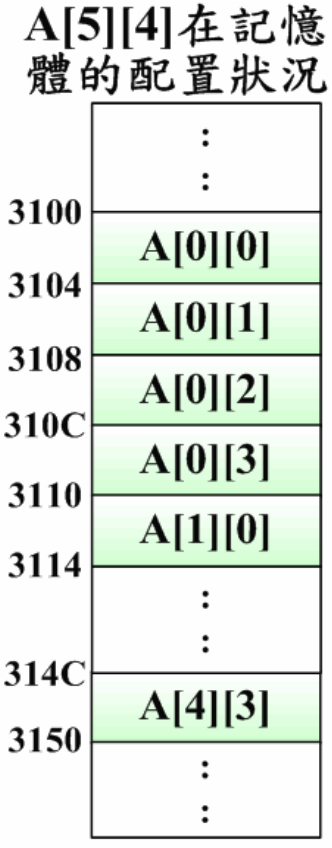
6.1.4　二維陣列

- 陣列若具有兩個索引稱爲『二維陣列』。
- 二維陣列的使用十分廣泛（僅次於一維陣列）。您可以將二維陣列以數學之矩陣來加以看待，也就是二維陣列是由『列（Row）』與『行（Column）』組合而成。
- 每一個元素恰恰落在特定之某一列的某一行。
- 釐清所謂的列與行的口訣：所謂『列』，指的是『橫列』，而『行』指的是『直行』。

6.1.4 二維陣列

- 『列』也就是二維陣列的第一維索引，而『行』則是二維陣列的第二維索引，我們以下圖來解說整數A[5][4]二維陣列在記憶體中的儲存。





6.1.4 二維陣列

- 由上述可知我們可以用二維陣列來表示複雜的資料，例如倘若使用橫列來表示各分公司的營運狀況，直行表示各季的營業額，則可以如下圖安排整間公司的總體營運狀況。

	第一季	第二季	第三季
台北總公司（第1列）	A[0][0]	A[0][1]	A[0][2]
新竹園區（第2列）	A[1][0]	A[1][1]	A[1][2]
高雄分公司（第3列）	A[2][0]	A[2][1]	A[2][2]

6.1.4 二維陣列

- 二維陣列宣告，可以使用列與行來分別代表兩個索引，每個索引長度（維度的元素個數）都必須填入[ ]之中，宣告如下：

語法：資料型態　陣列名稱[列的大小][行的大小];
功能：宣告二維陣列，以及元素的資料型態。

- 完整的二維陣列宣告語法如下：

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

  - 在上面的營運業績範例，A[3][4]陣列共有3列、4行，包含(3*4)=12個元素，若要取得高雄分公司第3季的營業額，則應該以相對應的索引值來加以取得，也就是A[2][2]。（在C語言中，二維陣列的行列索引起始值仍是由0開始計算）

6.1.4 二維陣列

- 【實用範例6-5】：將九九乘法表的乘法結果儲存在9x9的二維整數陣列之中，並將陣列的資料列印出來。
- 範例6-5：ch6_05.c

```
1  /*****************************
2       檔名:ch6_05.c
3       功能:二維陣列的練習
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  void main(void)
8  {
9   int m[9][9];
10  int i,j;
```

6.1.4 二維陣列

```
11
12  for(i=1;i<=9;i++)
13      for(j=1;j<=9;j++)
14          m[i-1][j-1]=i*j;
15    for(i=1;i<=9;i++)
16   {
17      for(j=1;j<=9;j++)
18      {
19          printf("%d*%d=%d\t",i,j,m[i-1][j-1]);
20      }
21      printf("\n");
22   }
23   /*  system("pause");  */
24  }
25
```

6.1.4 二維陣列

- 執行結果：

```
1*1=1  1*2=2  1*3=3  1*4=4  1*5=5  1*6=6  1*7=7  1*8=8  1*9=9
2*1=2  2*2=4  2*3=6  2*4=8  2*5=10 2*6=12 2*7=14 2*8=16 2*9=18
3*1=3  3*2=6  3*3=9  3*4=12 3*5=15 3*6=18 3*7=21 3*8=24 3*9=27
4*1=4  4*2=8  4*3=12 4*4=16 4*5=20 4*6=24 4*7=28 4*8=32 4*9=36
5*1=5  5*2=10 5*3=15 5*4=20 5*5=25 5*6=30 5*7=35 5*8=40 5*9=45
6*1=6  6*2=12 6*3=18 6*4=24 6*5=30 6*6=36 6*7=42 6*8=48 6*9=54
7*1=7  7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49 7*8=56 7*9=63
8*1=8  8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64 8*9=72
9*1=9  9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```

6.1.4 二維陣列

- 二維陣列初始化
  - 在宣告二維陣列的同時，也可以指定陣列元素的初始值（和一維陣列類似），語法如下：

語法：資料型態 陣列名稱[列的大小][行的大小]= {{M_{00},M_{01},M_{02},…,M_{0j-1}},
{M_{10},M_{11},M_{12},…,M_{1j-1}},
:
:
{$M_{(i-1)0}$,$M_{(i-1)1}$,$M_{(i-1)2}$,…,$M_{(i-1)(j-1)}$}
};

功能：宣告二維陣列並設定陣列元素的初始值。

6.1.4 二維陣列

- 【語法說明】：
  - 陣列元素必須依序指定每一列元素的內容，而列元素的內容則將之以{ }包裝起來。

```
int score[5][3]={ {85,78,65},
                  {75,85,69},
                  {63,67,95},
                  {94,92,88},
                  {74,65,73} };
```

  - 根據初始值的個數，自行判斷該陣列的元素個數，即上例也可以如下宣告：

```
int score[][] = { {85,78,65},
                  {75,85,69},
                  {63,67,95},
                  {94,92,88},
                  {74,65,73} };
```

6.1.4 二維陣列

- 當明確定義陣列大小時，若設定的陣列元素初始值不足，則會將剩餘未設定初值的元素內容設定爲0（針對數值陣列而言）。
  - 【範例】：

```
int score[5][3]={0};
```

- 【觀念範例6-6】：使用二維陣列存放學生的期中考成績。
- 範例6-6：ch6_06.c

```
1  /*****************************
2       檔名:ch6_06.c
3       功能:二維陣列的練習
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
```

6.1.4 二維陣列

```
 8  void main(void)
 9  {
10   float score[5][4] = { {85,78,65,0},
11                         {75,85,69,0},
12                         {63,67,95,0},
13                         {94,92,88,0},
14                         {74,65,73,0} };
15
16   int i,j;
17   printf("計概\t數學\t英文\t平均\n");
18   printf("============================\n");
19   for(i=0;i<5;i++)
20   {
21     score[i][3] = (score[i][0]+score[i][1]+score[i][2])/3;
22     for(j=0;j<4;j++)
23     {
24         printf("%.2f\t",score[i][j]);
25     }
26     printf("\n");
27   }
28   /*  system("pause");  */
29  }
```

6.1.4 二維陣列

- 執行結果：

```
計概      數學      英文      平均
=============================
85.00     78.00     65.00     76.00
75.00     85.00     69.00     76.33
63.00     67.00     95.00     75.00
94.00     92.00     88.00     91.33
74.00     65.00     73.00     70.67
```

- 範例說明：
  - 陣列中每一列代表一個學生的成績，所以共有5位學生的成績。
  - 每一列的第4個元素（即score[i][3]）是用來存放該列的平均分數。

6.2 字串

- 在C語言中，字串其實就是一維的字元陣列，但是這個字元陣列有一個特殊的結尾元素「\0」字元)。

- 「\0」稱之爲空字元（null character），而一般的字元陣列則無此規定。換句話說，字串是一種字元陣列，但字元陣列不一定是字串。

- 由於字串是一種特殊的字元陣列，因此我們將字串稱之爲字元字串(character string)。

6.2.1 字元陣列與字串的宣告：

- 宣告一般的一維字元陣列範例如下：

```
char
string1[]={'W','e','l','c','o','m','e'};
```

- 宣告字串（即字元字串；特殊的字元陣列）範例如下：

```
char string2[]="Welcome";
```

一般字元陣列與字串（特殊字元陣列）的區別

6.2.1 字元陣列與字串的宣告：

- 經過上述兩種宣告之後，字元陣列與字串的記憶體配置如上圖（您可以使用sizeof運算子分別求出字元陣列與字串的記憶體大小，就會發現兩者所佔用的記憶體空間剛剛好相差1個byte），其中宣告字串時的結尾字元「\0」，編譯器會自動產生，不需要由我們宣告，但是我們也可以在宣告字元陣列時，手動填上「\0」結尾字元，如此一來，該字元陣列就可以被視爲字串了，如下範例。

```
char string3[]={'W','e','l','c','o','m','e','\0'};     /*
string3可視爲字串變數  */
```

6.2.2 字串陣列的宣告

- 宣告字串陣列很簡單，只需要直接指定各個陣列元素（字串）的初始值即可（字串陣列其實是二維陣列），但必須要注意的是，字串的最大長度（即第二維度的長度）必須明確宣告，而第一維度的長度則可由編譯器自動計算，如下範例。

```
char StringArray[ ][6]
={"human","dog","cat","bird"};
```

	[0]	[1]	[2]	[3]	[4]
StringArray[0]	h	u	m	a	n
StringArray[1]	d	o	g	\0	$
StringArray[2]	c	a	t	\0	$
StringArray[3]	b	i	r	d	\0

6.2.3 字串整合範例

- 【觀念範例6-7】：計算字串長度（字元個數）。
- 範例6-7：ch6_07.c

```
1  /*****************************
2      檔名:ch6_07.c
3      功能:計算字串長度
4   *****************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7   void main(void)
8  {
9   char string[]="I love Kitty";
10  int i;
11   i=0;
12  while (string[i]!='\0')
13  {
14    i++;
15  }
16  printf("字串%s的長度爲%d\n",string,i);
17  /*  system("pause");  */
18 }
```

- 執行結果：

```
字串I love Kitty的長度爲12
```

6.2.3 字串整合範例

- 範例說明：
  - 程式執行完畢，i值爲12，也就是string[12]='\0'。
  - 由於代表字串的特殊字元陣列由索引『0』開始記錄，一直記錄到索引『11』恰爲『I love Kitty』，因此字串長度爲12（代表字串的特殊字元陣列則必須佔用13個bytes）。
  - 字串長度是程式設計師時常要關心的數值，由自己親自設計程式計算字串長度是一個辦法，但並不是一個好辦法，這是因爲C語言提供了方便的計算字串長度函式（在下一節中將會介紹的strlen()函式）。

6.2.3 字串整合範例

- 【觀念範例6-8】：將字串反向（即所有的字元倒過來）。
- 範例6-8：ch6_08.c

```
1  /*****************************
2      檔名:ch6_08.c
3      功能:字串反向
4   *****************************/
5
6  #include <stdio.h>
7  #include <stdlib.h>
8
9  void main(void)
10 {
11  char string1[60],string2[60];
12  int i,len;
13
14  printf("請輸入字串:");
15  scanf("%s",&string1);
```

6.2.3 字串整合範例

```
17   len=0;
18   while(string1[len]!='\0')
19   {
20      len++;
21   }
22   for(i=0;i<len;i++)
23   {
24      string2[i]=string1[len-1-i];
25   }
26   string2[i]='\0';
27
28   printf("反向字串爲:%s\n",string2);
29   /*  system("pause");  */
30  }
```

```
請輸入字串:Welcome
反向字串爲:emocleW
```

6.2.3 字串整合範例

- 範例說明：
  - 第18~21行：計算字串長度，len代表字串長度，若輸入Welcome字串，則len爲7。
  - 第22~25行：手動將字串string1的每個字元反向填入string2的字串陣列中，但最後的結尾字元「\0」則不填。
  - 第26行：手動將string2字串加上結尾字元「\0」。
  - 由這個範例可以得知，我們可以手動將字元字串的內容一一填入陣列中，完成一個字串（但千萬要記得填入結尾字元'\0'）。
  - 第11行：宣告字串（陣列模式），但不給予初值，此時必須提供陣列大小（本例爲60）。
  - 本範例使用scanf()讀取單一輸入字串，輸入字串中不可包含空白字元。

6.3 字串相關函式

- C語言提供了與字串處理有關的函式，使得程式設計師在字串處理時免去許多麻煩，我們將在這一節中選擇其中幾個重要且常用的函式來加以說明，其餘未說明的函式，則請參閱『C/C++函式庫』專書或網路查尋。

6.3.1 輸入與輸出－gets( )、fgets( )與puts( )

- 在前面我們提到過的C語言輸入函式有scanf( )與printf( )，這兩個函式可以用來輸入與輸出字串。不過，scanf( )所讀取的輸入字串，中間不可以出現空白字元，使用gets( )、fgets( )函式就可以解決了。

- 同樣地，printf( )並不會在輸出字串後自動換行（必須使用\n跳脫字元），此時，您只要改用puts( )函式來輸出字串，就會自動換行了。

6.3.1 輸入與輸出－gets( )、fgets( )與puts( )

- gets( )

標頭檔：#include <stdio.h>
語法：char *gets(char *s);
功能：從標準輸入裝置（鍵盤）讀取一整行字串。

- 【語法說明】：
  - (1) gets( )會從輸入裝置讀取一個字串(直到遇到「\n」換行字元爲止)，並且存放到指定的指標字串s中，同時會自動在字串結尾加上「\0」符號。
  - (2) gets( )與scanf( )最大的不同處在於，scanf( )在讀取字串時，若遇空白字元或是「\n」換行符號便判定字串結束，但gets( )卻可以接受字串中出現空白字元；換句話說，scanf( )只能讀取不含空白字元的字串，而gets( )卻無此限制。

6.3.1 輸入與輸出－gets( )、fgets( )與puts( )

- fgets( )

標頭檔：#include <stdio.h>
語法：char *fgets(char *s, int size, FILE *stream);
功能：從指定的檔案中讀取一整行字串（包含換行字元）
或讀取size大小的字串。

- 【語法說明】：
  - (1) fgets( )主要是用來取出檔案中的字串，但由於gets( )有部分問題，所以一般我們都會使用fgets( )來加以取代。如果我們將檔案引數stream指定爲stdin（stardand input；標準輸入裝置；即鍵盤），fgets( )就可以取代gets( )了。
  - (2) fgets( )同樣允許字串中出現空白字元，並且也會將最後一個換行字元存放到字串s中，因此會出現換行效果。

6.3.1 輸入與輸出－gets( )、fgets( )與puts( )

- 範例6-9：ch6_09.c

```
1   /*************************
2       檔名:ch6_09.c
3       功能:fgets()函式練習
4    *************************/
5   #include <stdio.h>
6   #include <stdlib.h>
7    void main(void)
8   {
9    char fgets_string[100];
10   char scanf_string[100];
11   printf("請輸入fgets字串:");
12   fgets(fgets_string,100,stdin);
13   printf("請輸入scanf字串:");
14   scanf("%s",&scanf_string);
15    printf("您輸入的fgets字串是%s",fgets_string);
16   printf("您輸入的scanf字串是%s\n",scanf_string);
17   /*  system("pause");  */
18  }
```

6.3.1 輸入與輸出－gets( )、fgets( )與puts( )

- 執行結果：

```
請輸入fgets字串:This is a book
請輸入scanf字串:This is a book
您輸入的fgets字串是This is a book
您輸入的scanf字串是This
```

6.3.1 輸入與輸出－gets( )、fgets( )與puts( )

- 範例說明：
  - 第15行：透過fgets( )讀取字串並存入fgets_string。
  - 由於fgets( )設定檔案爲stdin標準輸入裝置，因此會從鍵盤讀取資料。
  - 第17行：使用scanf( )讀取字串。
  - (3) 在執行結果中，您可以發現同樣輸入『This is a book』，但scanf( )卻只會讀取到『This。
  - (4) 第19行中，我們並未加入換行符號（\n），但在輸出結果中，卻可以發現仍然出現了換行現象，這是因爲fgets( )會將最後的換行符號也一併存入fgets_string的緣故。

6.3.1 輸入與輸出－gets( )、fgets( )與puts( )

- puts( )

標頭檔：#include <stdio.h>
語法：int puts(const char *s);
功能：輸出字串s到標準輸出裝置。輸出字串完畢後，將自動換行。

- 【語法說明】：
  - puts( )函式除了會將字串輸出到標準輸出裝置stdout（預設爲螢幕）之外，還會輸出一個換行符號，產生自動換行的效果。

6.3.1 輸入與輸出－gets( )、fgets( )與puts( )

- 範例6-11：ch6_11.c

```
1  /*************************
2      檔名:ch6_11.c
3      功能:puts()函式練習
4   *************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7   void main(void)
8  {
9    char string1[100],string2[100];
10   printf("請輸入string1字串:");
11   fgets(string1,100,stdin);
12   printf("請輸入string2字串:");
13   scanf("%s",&string2);
14   printf("string1字串是");
15   puts(string1);
16   printf("string2字串是");
17   puts(string2);
18   /*  system("pause");  */
19 }
```

6.3.1 輸入與輸出－gets( )、fgets( )與puts( )

- 執行結果：

```
請輸入string1字串:Welcome
請輸入string2字串:Welcome
string1字串是Welcome

string2字串是Welcome
```

- 範例說明：
  - 第14行：透過fgets( )讀取字串並存入string1。string1的內容爲『Welcome**LF**』。（**LF**爲換行字元）
  - 第16行：使用scanf讀取字串並存入string2。string2的內容爲『Welcome』，不含換行字元。

6.3.1 輸入與輸出－gets( )、fgets( )與puts( )

- 範例說明：
  - 在執行結果中，『string1字串是Welcome』的換行效果是因爲string1的『**LF**』造成的效果；接下來的空白行則是由於puts( )會自動換行的緣故，所以有兩次的換行效果。最後輸出的『string2字串是Welcome』之後也會換行，這是puts( )的自動換行，所以只有一次的換行效果。

6.3.2　計算字串長度－strlen( )

- 在前面的範例中，我們曾撰寫程式來計算字串的長度，其實我們根本不用如此做，因爲有現成的strlen( )函式可以使用，並且已被納入ANSI C的標準函式庫string.h之中。
- strlen( )

標頭檔：#include <string.h>
語法：size_t strlen(const char *s);
功能：計算字串s的長度，字串長度即字元個數（不包含字串結尾'\0'）。

  - 【語法說明】：
    - 假設字串s爲"Welcome"、則使用strlen(s)將回傳整數7。

6.3.2 計算字串長度－strlen( )

- 【實用範例】：使用strlen( )函式改寫範例6-8，製作反向字串並將之輸出。
- 範例6-12：ch6_12.c。

```
1  /*******************************
2      檔名:ch6_12.c
3      功能:strlen()練習-字串反向
4   *******************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8   void main(void)
9  {
10  char string1[60],string2[60];
11  int i,len;
12   printf("請輸入字串:");
13  scanf("%s",&string1);
```

6.3.2 計算字串長度－strlen( )

```
14
15
16
17
18
19
20
21
22
23
```

```
 len=strlen(string1);
 printf("字串長度爲%d\n",len);
  for (i=0;i<len;i++)
 {
     string2[i]=string1[len-1-i];
 }
 string2[i]='\0';
   printf("反向字串爲:%s\n",string2);
 /*  system("pause");  */
}
```

執行結果：

```
請輸入字串:Welcome
字串長度爲7
反向字串爲:emocleW
```

6.3.2 計算字串長度－strlen( )

- 範例說明：
  - (1) 第8行：使用strlen( )函式，必須引入標頭檔string.h。
  - (2) 第19行：len將會是字串string1的長度。（string1="Welcome"、則len=7）
  - (3) 第22~26行：製作反向字串string2。

6.3.3複製字串－strcpy( )與strncpy( )

- string.h中也提供了複製字串的函式strcpy( )與strncpy( )，兩者的差別則在於複製全部字串（strcpy）或複製部分字串（strncpy）。
- strcpy( )

標頭檔：#include <string.h>
語法：char *strcpy(char *dest, const char *src);
功能：將src字串內容複製到dest字串中。

- strncpy( )

標頭檔：#include <string.h>
語法：char *strncpy(char *dest, const char *src, size_t n);
功能：將src字串的前n個字元複製到dest字串中。

6.3.3複製字串－strcpy( )與strncpy( )

- 【觀念範例6-13】：複製字串與複製部分字串，並藉由範例認清中文字的長度。
- 範例6-13：ch6_13.c

```
1  /*************************************************
2      檔名:ch6_13.c
3      功能:strcpy(),strncpy()練習-複製字串及部分字串
4  *************************************************/
5   #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8   void main(void)
9  {
10   char src_string[]="程式設計C語言";
11   char dest_string[60],dest_substring[60];
12   strcpy(dest_string,src_string);
13   strncpy(dest_substring,src_string,8);
14   printf("複製的完整字串爲:%s\n",dest_string);
15   printf("複製的部分字串爲:%s\n",dest_substring);
16   /*  system("pause");  */
   }
```

6.3.3複製字串－strcpy( )與strncpy( )

- 執行結果：

```
複製的完整字串爲:程式設計C語言
複製的部分字串爲:程式設計
```

- 範例說明：
  - 第15行：使用strcpy( )函式，將src_string內容複製到dest_string。
  - 第16行：使用strncpy( )函式，將src_string內容的前8個字元複製到dest_string。由於一個中文字佔用兩個字元，因此只會複製『程式設計』四個中文字。

6.3.4字串連結－strcat( )與strncat( )

- strcat( )與strncat( )可以用來連結兩個字串，差別在於第二個字串的連結字元個數。
- strcat( )

標頭檔：#include <string.h>
語法：char *strcat(char *dest, const char *src);
功能：將src字串內容連結到dest字串的結尾。

- 【語法說明】：
  - (1) 把src字串連接到dest字串的後面，編譯器會先假設dest有足夠的空間可以存放新連結進來的字串。
  - (2) 由於dest原本的內容將被覆蓋（變成dest+src），因此如果不想要改變原始字串的內容則可以使用如下方法：

6.3.4字串連結－strcat( )與strncat( )

- strncat( )

標頭檔：#include <string.h>
語法：char *strncat(char *dest, const char *src, size_t n);
功能：將src字串的前n個字元連結到dest字串的字尾。

  - 【語法說明】：
    - strncat( )也能把兩個字串連結起來，而且還可以指定第二個字串要被連結的字元個數。
- 【實用範例6-14】：組合字串。將string1、string2、string3組合起來，其中string1、string3可以由使用者輸入決定。
- 範例6-14：ch6_14.c

6.3.4 字串連結－strcat( )與strncat( )

```
1  /*****************************************
2      檔名:ch6_14.c
3      功能:strcpy(),strcat( )練習-組合字串
4   *****************************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8   void main(void)
9  {
10  char string1[20],string2[20]="股份有限公司",string3[20];
11  char dest_string[60];
12  printf("請輸入string1字串:");
13  scanf("%s",&string1);
14  printf("請輸入string3字串:");
15  scanf("%s",&string3);
16  strcpy(dest_string,string1);
17  strcat(dest_string,string2);
18  strcat(dest_string,string3);
19  printf("dest_string:%s\n",dest_string);
20  /*  system("pause");  */
21 }
```

6.3.4字串連結－strcat( )與strncat( )

- 執行結果：

```
請輸入string1字串:ULBITA
請輸入string3字串:出版
dest_string: ULBITA股份有限公司出版
```

- 範例說明：
  - (1) 第20行：複製string1字串，作爲開頭子字串。
  - (2) 第21行：使用strcat( )函式，作字串連結。（dest_string將會是string1與string2的連接結果）。
  - (3) 第22行：再度使用strcat( )函式，作字串連結。（dest_string將會是string1、string2、string3的連接結果）。

6.3.5字串比較－strcmp( )與strncmp( )

- strcmp( )可以用來比較兩個字串內容是否相等，

標頭檔：#include <string.h>
語法：int strcmp(const char *s1, const char *s2);
功能：比較s1、s2字串是否相等。

  - 【語法說明】：
    - 當字串s1與s2內容相等時，回傳「0」。若s1<s2，回傳「負整數值」。若s1>s2，回傳「正整數值」。所謂『<』、『>』是根據字元的ASCII值來作比較，例如：『a』<『k』。
    - 並且會由第一個字元開始比較，若相等才會比較第2個字元，依此類推，直到比出大小或判定兩個字串相等。
- strncmp( )

標頭檔：#include <string.h>
語法：int strncmp(const char *s1, const char *s2, size_t n);
功能：比較s1、s2字串的前n個字元是否相等。

6.3.5字串比較－strcmp( )與strncmp( )

- 範例6-15：ch6_15.c

```
1  /*****************************************
2      檔名:ch6_15.c
3      功能:strcmp()與strncmp()練習-比較字串
4   *****************************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8   void main(void)
9  {
10   char s1[]="output";
11   char s2[]="outside";
12   char s3[]="output";
13   char s4[]="outlook";
14   printf("%s與%s\t的比較結果==>%d\n",s1,s1,strcmp(s1,s1));
15   printf("%s與%s\t的比較結果==>%d\n",s1,s2,strcmp(s1,s2));
16   printf("%s與%s\t的比較結果==>%d\n",s1,s3,strcmp(s1,s3));
17   printf("%s與%s\t的比較結果==>%d\n",s1,s4,strcmp(s1,s4));
18   printf("%s與%s\t的前3個字元比較結果==>%d\n",s1,s2,strncmp(s1,s2,3));
19   /*  system("pause");  */
20 }
```

6.3.5字串比較－strcmp( )與strncmp( )

- 執行結果：

```
output與output   的比較結果==>0
output與outside  的比較結果==>-1
output與output   的比較結果==>0
output與outlook  的比較結果==>1
output與outside  的前3個字元比較結果==>0
```

- 範例說明：
  - 第17行：自己與自己相比，當然相同，所以結果爲0。
  - 第18行：前面的『out』比不出結果，而『p』的ASCII值 < 『s』的ASCII值，所以回傳負值（-1）。
  - 第19行：由於s1內容『output』與s3內容『output』相同，所以回傳0。
  - 第20行：前面的『out』比不出結果，而『p』的ASCII值 > 『l』的ASCII值，所以回傳正值（1）。
  - 第21行：前面3個字元都是『out』，所以回傳0。

6.3.6 句元分割－strtok( )

- 在程式應用的範疇中，將字串分割成一個個的元素是常常見到的應用，例如：一次輸入多筆資料，並使用『:』加以分隔。

- 此時我們就可以透過strtok( )函式，將字串依照某些分隔字元拆成一小段一小段的元素。

6.3.6 句元分割－strtok( )

- strtok( )

標頭檔：#include <string.h>
語法：char *strtok(char *s, const char *delim);
功能：delim為分隔字元，s為欲切割的字串來源。

- 【語法說明】：
  - strtok( )函式會將s字串中所出現的特殊字元（在delim字串中指定這些字元）當作分隔符號，將字串s切割成許多的token並一一回傳這些token，直到遇見指標字串的NULL爲止
  - delim可以指定爲一個以上的字元。這些字元都會被當成分割符號，並使用『,』連結每一個分隔字元。
  - 回傳值必須是指標字串。

6.3.6 句元分割－strtok( )

- 範例6-16：ch6_16.c。

```
1  /*******************************
2      檔名:ch6_16.c
3      功能:strtok()練習-取出token
4   *******************************/
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8   void main(void)
9  {
10  char string1[]="Word:Excel:PowerPointer:Access;C;Java;ASP:PHP";
11  char deliml[]=":,;";
12  char *Token;/* 指標字串,請見指標章節說明   */
13  printf("原始字串爲%s\n",string1);
14  printf("開始切割.....\n");
15  printf("Tokens(句元)如下:\n");
16  Token = strtok(string1,deliml); /* 將第一個句元存入Token */
17  while(Token != NULL)            /* 使用迴圈取出剩餘句元 */
18  {
19     printf("%s\n",Token);
20     Token = strtok(NULL,deliml);
21  }
22 }
```

6.3.6 句元分割－strtok( )

- 執行結果：

```
原始字串爲
Word:Excel:PowerPointer:Access;C;Java;ASP:PHP
開始切割.....
Tokens(句元)如下:
Word
Excel
PowerPointer
Access
C
Java
ASP
PHP
```

- 範例說明：
  - 第13行：設定分隔字元『:』與『;』。
  - 第14行：宣告指標字串，請參照8.4.2一節。
  - 第19行：取出第一個句元。
  - 第20~24行：取出剩餘的句元。

6.4 本章回顧

- 『陣列』是一種非常重要的資料結構，幾乎各種高階程式語言都支援『陣列』資料結構。C語言也提供了『陣列』。
- C語言陣列的索引值由0開始計算。
- 使用陣列可以免除大量變數命名的問題，使得程式具有較高的可讀性。
- 陣列將會佔用連續的記憶體空間。

6.4 本章回顧

- 陣列可以分爲一維陣列、二維陣列...等等。
- 陣列必須經由宣告，然後再透過索引存取陣列元素。
- 我們在宣告陣列的同時，也可以指定陣列元素的初始值。

語法：資料型態 陣列名稱[元素個數]={元素1初始值,元素2初始值,...};
功能：宣告一維陣列並設定陣列元素的初始值。

6.4 本章回顧

- 『字串』其實就是一維的字元陣列，但是這個陣列有一個特殊的結尾元素「\0」字元，因此我們將這種字串稱之爲字元字串(character string)。
- 宣告字串範例如下：
  - char string2[]="Welcome";

6.4 本章回顧

- 宣告字串陣列（特殊二維字元陣列）很簡單，只需要直接指定各個陣列元素（字串）的初始值即可，但該特殊字元陣列的最大長度（即第二維度的長度）必須宣告，而第一維度的長度則可由編譯器自動計算，如下範例。
  - char StringArray[ ][6] ={"human","dog","cat","bird"};

6.4 本章回顧

函式	標頭檔	功能
gets( )	stdio.h	從標準輸入裝置讀取一整行字串。
fgets( )		從指定的檔案中讀取一整行字串（包含換行字元）或讀取指定長度的字串。
puts( )		輸出字串到標準輸出裝置。輸出字串完畢後，將自動換行。
strlen( )	string.h	計算字串的長度，字串長度即字元個數（不包含字串結尾'\0'）。
strcpy( )		複製字串內容。
strncpy( )		複製字串的子字串（前n個字元）內容。
strcat		連結字串內容。
strncat( )		連結字串的子字串（前n個字元）內容。
strcmp( )		比較兩字串是否相等。
strncmp( )		比較兩字串的前n個字元是否相等。
strtok( )		切割字串爲一個個的token。

6.4 本章回顧

- 『搜尋』與『排序』是程式設計的一項基本且重要的問題。所謂『搜尋』（Searching），指的是在一堆資料中，尋找您所想要的資料。
- 所謂『排序』（Sorting）則是將一堆雜亂的資料，依照某個鍵值（Key Value）依序排列，方便日後的查詢或使用。在本章中，我們補充介紹了『氣泡排序法』。

本章習題