Example 25

檔案處理：二進位檔

CTM 公司舉辦新產品說明會，現在要預訂說明會當天中午的便當。寫一程式調查當天參加者中午用餐的情形；輸入的資料包含：姓名、電話、葷食或素食，葷食有 2 種便當可供選擇：雞腿與排骨便當；調查資料以結構表示並以二進位檔案儲存。

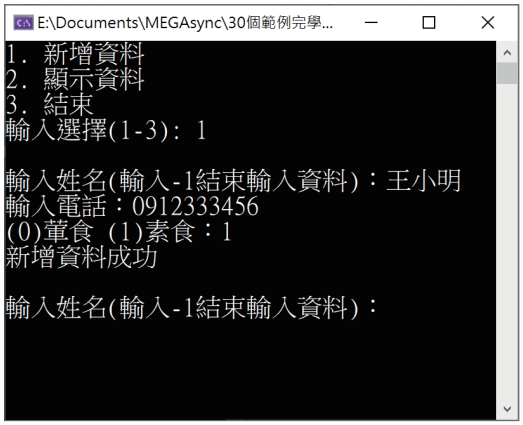
一、學習目標

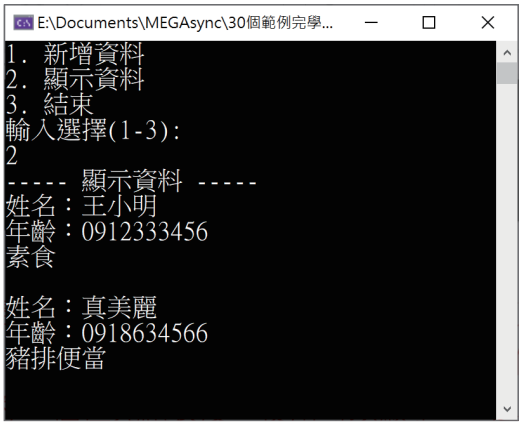
二進位檔案與文字檔案最主要的差別，在於對資料儲存的方式是以二進位的值儲存；因此使用一般的文書軟體開啓二進位的檔案時，其內容並無法理解與解讀。無法以文字形式儲存的資料，則適合以二進位檔案的方式儲存；例如：音樂、影像等。

本範例以單個變數、陣列與結構此 3 種不同類型的資料，示範如何將資料儲存爲二進位檔案，以及如何從二進位檔案正確讀取這些資料。

二、執行結果

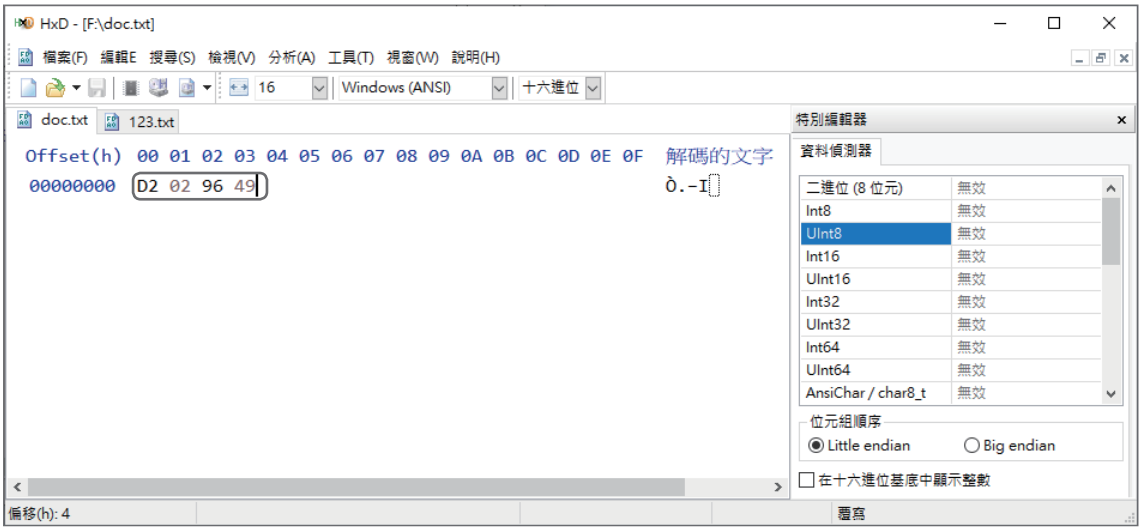
程式執行後會顯示選單：「1. 新增資料」、「2. 顯示資料」與「3. 結束」，並讓使用者輸入選項編號。下圖左爲新增資料畫面：每次新增一筆資料後會立即將資料存檔。下圖右爲顯示資料畫面：開啓檔案並從檔案中逐筆讀取並顯示資料。



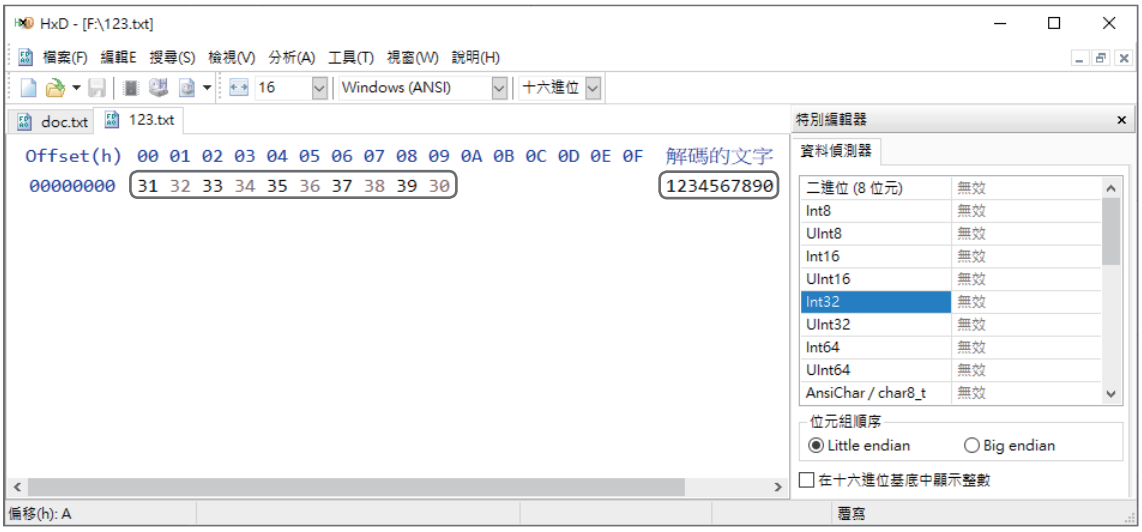


二進位檔與一般的文字檔的主要差別，除了能儲存無法以文字表達的資料之外，對於數值的儲存方式也不相同。

例如：有一個 `int` 型別的整數 `num`，其值等於 1234567890，使用二進位檔儲存此值只會佔用 4 個位元組；因爲 `int` 資料型別的長度等於 4 個位元組，任何 `int` 型別的數值儲存於檔案之後，都只會佔用固定的 4 個位元組長度。使用可以顯示 16 進位的軟體開啓檔案後，可以看到如下的資料：D2029649 即爲 1234567890 的 16 進位數值，的確佔用了 4 個位元組。



此數值若以文字檔的形式儲存，則數值中每個數字佔 1 個位元組，因此佔用 10 個位元組，並且此數值是以文字的方式儲存；如下圖所示。例如：第 1 個位元組資料 31 即是文字 1 的 ASCII 碼的 16 進位數值。



二進位檔仍然使用 fstream、ifstream 與 ifstream 類別宣告檔案變數，但開檔模式一定會有 ios::binary 常數，再搭配其他的開檔模式常數；例如：

```
1 fstream file;
2
3 file.open("doc.txt", ios::binary | ios::out);
4 file.open("doc.txt", ios::binary | ios::out|ios::app);
5
6 file.close();
```

程式碼第 3 行使用的開檔模式爲 ios::binary|ios::out；因此，此檔案爲供寫入的二進位檔案。第 4 行的開檔模式爲 ios::binary|ios::out|ios::app；因此，此檔案爲爲供寫入的二進位檔案，並且會將寫入的資料附加於檔尾。

25-1　儲存與讀取基本型別資料

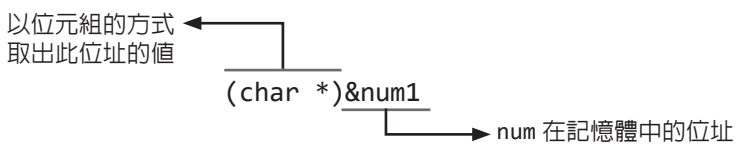
讀寫字元與數值

假設二進位檔案變數爲 file，則將 int 型別的變數 num 寫入檔案的方式如下所式：

```
1 int num = 12;
2
3 file.write((char *)&num, sizeof(int));
```

寫入 4 個位元組

因爲變數 num 爲 int 型別，而 int 型別是固定的長度（4 個位元組）；因此，write() 的第 2 個引數 sizeof(int) 讓 write() 函式知道要寫入 4 個位元組的資料。而寫入檔案的資料來源爲變數 num 在記憶體中的位址（&num）開始，從此位址取出 4 個位元組的內容（char *）寫入檔案，因此函式 write() 的第 1 個引數才會是如下的寫法：



要從檔案中讀取數值資料也是相同的方式；如下所示。從檔案變數 file 中讀取 4 個字元，並以位元組的型式放入變數 num 在記憶體中的位址。

```
1 int num;
2
3 file.read((char *)&num, sizeof(int));
```

讀取 4 個位元組

讀寫字元的方式也相同，如下所示：

```
1 char ch = 'A';
2
3 file.write((char *)&ch, sizeof(char));
```

因爲變數 ch 本身已經是字元型別，程式碼第 3 行不用再強調以字元的形式寫入檔案，所以可以簡化爲：

```
3 file.write(&ch, sizeof(char));
```

從檔案中讀取字元也是相同的方式，則如下所示：

```
1 char ch;
2
3 file.read(&ch, sizeof(char));
```

讀寫陣列資料

陣列資料大致上可區分爲數值資料與字元資料此 2 種形式；例如：整數陣列、浮點數陣列、雙精度浮點數陣列等，都是屬於數值型別的陣列。字串是由字元陣列所組成，因此屬於字元型別的資料；至於 string 型別的字串，通常會先轉換爲字元陣列形式的字串之後，再寫入檔案。

▶ 數值陣列

例如：有一整數型別的一維陣列 arr[3]，其元素爲 11,12,13；則將此陣列寫入二進位檔案 file 的方式如下所示。int 型別的資料長度等於 4 個位元組，所以此陣列實際佔有的空間爲 12 位元組。sizeof(arr) 能取得陣列 arr 的大小，因此將 sizeof(arr) 作爲函式 write() 的第 2 個引數，便能知道要從陣列 arr 的起始位址開始取出 12 個位元組寫入檔案。

```
1 int arr[3] = { 11,12,13 };
2
3 file.write((char*)&arr, sizeof(arr));
```

除了使用 sizeof(arr) 取得陣列大小，也能使用 sizeof(int)×3 計算陣列的大小：陣列 arr 爲 int 型別，int 型別的長度等於 4。陣列 arr 有 3 個元素；因此陣列的大小便是 4×3=12。

```
sizeof(int) × 3 = 4 × 3 = 12
```

從檔案讀取資料並儲存到數值陣列，如下所示。

```
1 int arr[3];
2
3 file.read((char*)&arr, sizeof(arr));
```

▶ 字元陣列

假設有字元型別的字串，則將此字串儲存到檔案與從檔案中讀取的方式，如下所示。程式碼第 3 行將字串 str1 寫入檔案，第 5 行從檔案讀取資料並儲存到陣列 arr2。

```
1 char str1[10] = "王小明",str2[10];
2
3 file.write(str1, sizeof(str1));
4     ⋮
5 file.read(str2, sizeof(str2));
```

須注意 write() 與 read() 函式的第 2 個引數使用的是 sizeof() 而不是 strlen()，否則存入檔案的字串資料與從檔案讀取的資料都會不正確。也就是說，使用字元陣列形式的字串，存入檔案的資料是整個陣列，而不是只有字串長度的資料而已；因此，從檔案讀取資料的長度也必須是整個陣列的長度。

▶ string 型別的字串

string 型別的字串由於可隨意變更字串內容，因此字串長度並不固定，所以不適合直接寫入檔案；因爲當要從檔案裡讀取資料時，無法知道原始字串的正確長度。所以 string 型別的字串通常會先轉換爲字元型別的字串之後，再寫入檔案；如下所示。

```
1 string str1 = "眞美麗";
2 char str2[20],str3[20];
3
4 sprintf_s(str2, "%s", str1.c_str())
5 file.write(str2, sizeof(str2));
6     ⋮
7 file.read(str3, sizeof(str3));
```

程式碼第 4 行使用 sprintf_s() 函式將字串 str1 轉型爲字元陣列形式的字串，並儲存於字元陣列 str2；須注意陣列 str2 的長度需要可以容納字串 str1 的內容，第 5 行再使用 write() 函式將字串陣列 str2 寫入檔案。第 7 行使用一般讀取字元陣列的方式，從檔案讀取將原來的字串資料並儲存於陣列 str3。

練習 1：讀寫二進位檔案

將字元 'A'、浮點數陣列 {3.14, 1.23, 2.56} 與字串 " 早安，你好。" 寫入二進位檔案。接著開啓此二進位檔案讀取並顯示這些資料。

解說

要開啓或建立二進位檔案，開檔模式使用 ios::binaey。假設二進位檔案變數爲 file，字元變數爲 ch，浮點數陣列變數爲 ft[3]，字元陣列形式的字串變數爲 str，則將這些資料寫入檔案的形式應如下所示。

```
1 file.write((char *)&ch, sizeof(ch));
2 file.write((char*)& ft, sizeof(ft));
3 file.write(str, sizeof(str));
```

而從檔案中讀取資料，並儲存於相對應的變數，則如下所示。

```
1 file.read((char *)&ch, sizeof(ch));
2 file.read((char*)& ft, sizeof(ft));
3 file.read(str, sizeof(str));
```

執行結果

```
寫入資料 ...OK.
讀取資料 ...
A
3.14 1.23 2.56
早安，你好。
```

程式碼列表

```
 1 #include <iostream>
 2 #include <fstream>
 3 using namespace std;
 4
 5 int main()
 6 {
 7     fstream file;
 8     char ch1 = 'A',ch2;
 9     float ft1[3] = { 3.14,1.23, 2.56 },ft2[3];
10     char str1[] = " 早安，你好。", str2[80];
11
```

```
12      file.open("doc.txt", ios::binary | ios::out);
13      if (!file)
14      {
15          cout << " 無法建立檔案。" << endl;
16          exit(0);
17      }
18
19      cout << " 寫入資料 ...";
20      file.write((char *)&ch1, sizeof(ch1));
21      file.write((char*)& ft1, sizeof(ft1));
22      file.write(str1, sizeof(str1));
23      file.close();
24      cout << "OK." << endl;
25
26      cout << " 讀取資料 ..." << endl;
27      file.open("doc.txt", ios::binary | ios::in);
28      file.read((char*)& ch2, sizeof(ch2));
29      file.read((char*)& ft2, sizeof(ft2));
30      file.read(str2, sizeof(str2));
31      file.close();
32
33      cout << ch2 << endl;
34      for (auto item : ft2)
35          cout << item << " ";
36      cout << endl;
37      cout << str2 << endl;
38
39      system("pause");
40  }
```

程式講解

1. 程式碼第 1-3 行引入需要的標頭檔與宣告使用 `std` 命名空間。
2. 程式碼第 7-10 行宣告變數。字元變數 `ch1`、浮點數陣列 `ft1` 與字串變數 `str1` 作爲寫入檔案的資料；字元變數 `ch2`、浮點數陣列 `ft2` 與字串變數 `str2`，則使用於儲存從檔案讀取的資料。字串變數 `str2` 的長度設定爲 `80`，這是假設並不知道從檔案讀取的字串資料有多長，所以字串變數 `str2` 只好預設比較大的長度。
3. 程式碼第 12 行使用開檔模式 `ios::binary` 與 `ios::out` 建立檔案 `doc.txt`；因此，這是一個供寫入資料的二進位檔案。第 13-17 行當建立檔案失敗則顯示訊息後結束程式。

4. 程式碼第 20-22 行使用 `write()` 函式分別將字元變數 `ch1`、浮點數陣列 `ft1` 與字串變數 `str1` 寫入檔案，第 23 行使用 `close()` 函式關閉檔案。
5. 程式碼第 27 行使用開檔模式 `ios::binary` 與 `ios::in` 開啓檔案 `doc.txt`；因此，這是一個供讀取資料的二進位檔案。
6. 程式碼第 28-30 行使用 `read()` 函式從檔案讀取資料並分別儲存於變數 `ch2`、浮點數陣列變數 `ft2` 與字串變數 `st2`。第 31 行使用 `close()` 函式關閉檔案。
7. 程式碼第 33-37 行分別顯示所讀取的資料；第 34-35 行使用 `for` 重複敘述顯示浮點數陣列 `ft2` 裡的 3 個元素。

25-2 儲存與讀取結構資料

結構資料可以直接寫入二進位檔案，也能從二進位檔案中讀取資料並儲存到結構。對於結構陣列則可以使用 `for` 或是 `while` 重複敘述，逐筆將結構陣列裡的資料寫入二進位檔案，也能一次把結構陣列的資料寫入檔案；從檔案讀出結構陣列的資料也是相同的方式。

讀寫結構

例如：有 1 個 `_MYSTRUCT` 結構型別的變數 `myStruct`，要將此結構變數寫入二進位檔案變數 `file`，則如下所示：

```
_MYSTRUCT myStruct
    ⋮
file.write((char*)&myStruct, sizeof(_MYSTRUCT));
```

若要從檔案中讀取結構型別的資料，則如下所示：

```
_MYSTRUCT myStruct
    ⋮
file.read((char*)&myStruct, sizeof(_MYSTRUCT));
```

讀寫結構陣列

若有 1 個長度等於 2 的結構陣列 `myStruct`，則使用 `for` 重複敘述將結構陣列資料寫入二進位檔案變數 `file` 的方式，如下所示。因爲結構陣列裡的每個元素都是 `_MYSTRUCT` 型別，所以寫入檔案的長度爲 `sizeof(_MYSTRUCT)`；寫入檔案的第 `i` 筆結構資料的起始位址則爲 `(char *)&myStruct[i]`。

```
_MYSTRUCT myStruct[2];
    ⋮
for (int i = 0; i < 2; i++)
{
    file.write((char*)&myStruct[i], sizeof(_MYSTRUCT));
          ⋮
}
```

若是要將結構陣列一次全部寫入檔案，則寫入檔案的長度則爲 `sizeof(_MYSTRUCT)×2`，寫入的結構陣列起始位址爲 `(char *)&myStruct[0]`；如下所示。

```
_MYSTRUCT myStruct[2];
    ⋮
file.write((char*)&myStruct[0], sizeof(_MYSTRUCT)*2);
```

要從檔案讀取結構陣列的資料，做法與寫入檔案的方式相同，只是把函式 `write()` 替換爲 `read`，以及開檔模式改爲 `ios::binary|ios::in`。

練習 2：使用結構讀寫二進位檔案

BMI 基本資料須填寫姓名、年齡、身高與體重。寫一程式，使用結構表示 BMI 基本資料，並新增 2 筆資料後寫入檔案；接著從檔案讀取資料並顯示資料。

解說

BMI 基本資料使用結構表示，可爲如下之形式。其中浮點數陣列 `data` 的長度等於 `2`，第 1 個元素 `data[0]` 用於儲存身高，第 2 個元素 `data[1]` 用於儲存體重。

```
struct _MYSTRUCT
{
    char name[20];  // 姓名
    int age;        // 年齡
    float data[2];  // 身高與體重
};
```

本練習題需要有以下之功能：新增資料、將資料寫入檔案、從檔案讀取資料與顯示資料。爲了避免主函式 `main()` 裡面撰寫太多程式碼而過於雜亂；因此可以將此 4 項功能寫成自訂函式，並在主函式 `main()` 中呼叫這些自訂函式。

執行結果

首先需要輸入 2 筆 BMI 基本資料，如下所示：

```
輸入姓名：王小明
輸入年齡：19
輸入身高 ( 公尺 ) 與體重 ( 公斤 )( 使用空白隔開 )：1.67 60.2
輸入姓名：眞美麗
輸入年齡：20
輸入身高 ( 公尺 ) 與體重 ( 公斤 )( 使用空白隔開 )：1.6 49.4
```

接著便會將資料寫入檔案，再從檔案讀取此 2 筆資料，最後顯示資料；若讀寫檔案沒有發生錯誤，則顯示如下之訊息：

```
----- 寫入資料 -----
----- 讀取資料 -----
----- 顯示資料 -----
```

從檔案讀取資料成功之後，則顯示所讀取的 2 筆 BMI 資料；如下所示：

```
姓名：王小明
年齡：19
身高：1.67
體重：60.2

姓名：眞美麗
年齡：20
身高：1.6
體重：49.4
```

程式碼列表

```
 1 #include <iostream>
 2 #include <fstream>
 3 using namespace std;
 4
 5 #define MAX_NUM 2  // 一共 2 筆資料
 6
 7 struct _MYSTRUCT
 8 {
 9     char name[20];  // 姓名
10     int age;         // 年齡
```

```
11      float data[2];  // 身高與體重
12  };
13
14  //------------- 填寫資料 ---------------------------
15  void addData(_MYSTRUCT myStruct[])
16  {
17      for (int i = 0; i < MAX_NUM; i++)
18      {
19          cout << " 輸入姓名：";
20          cin >> myStruct[i].name;
21          cout << " 輸入年齡：";
22          cin >> myStruct[i].age;
23          cout << " 輸入身高 ( 公尺 ) 與體重 ( 公斤 )( 使用空白隔開 )：";
24          cin >> myStruct[i].data[0] >> myStruct[i].data[1];
25      }
26  }
27
28  //------------- 寫入資料 ---------------------------
29  bool writeData(_MYSTRUCT myStruct[])
30  {
31      fstream file;
32      bool fg = true;
33
34      cout << "----- 寫入資料 -----" << endl;
35      file.open("doc.data",ios::binary | ios::out);
36      if (!file)
37          fg = false;
38       else
39       {
40          file.write((char*)& myStruct[0], sizeof(_MYSTRUCT)*MAX_NUM);
41          if(!file.good())
42              fg = false;
43          file.close();
44       }
45      return fg;
46  }
47  //------------- 讀取資料 ---------------------------
48  bool readData(_MYSTRUCT rStruct[])
49  {
50      fstream file;
51      bool fg = true;
52
```

```
53      cout << "----- 讀取資料 -----" << endl;
54      file.open("doc.data", ios::binary | ios::in);
55      if (!file)
56          fg = false;
57
58      for (int i = 0; i < MAX_NUM; i++)
59      {
60          file.read((char*)& rStruct[i], sizeof(_MYSTRUCT));
61          if (!file.good())
62          {
63              fg = false;
64              break;
65          }
66      }
67
68      file.close();
69      return fg;
70  }
71
72  //------------- 顯示資料 ----------------------------
73  void showData(_MYSTRUCT rStruct[])
74  {
75      cout << "----- 顯示資料 -----" << endl;
76      for (int i = 0; i < MAX_NUM; i++)
77      {
78          cout << "姓名：" << rStruct[i].name << endl;
79          cout << "年齡：" << rStruct[i].age << endl;
80          cout << "身高：" << rStruct[i].data[0] << endl;
81          cout << "體重：" << rStruct[i].data[1] << endl << endl;
82      }
83  }
84
85  int main()
86  {
87      _MYSTRUCT myStruct[MAX_NUM],rStruct[MAX_NUM];
88
89      addData(myStruct);  // 新增資料
90
91      if (!writeData(myStruct))  // 寫入資料
92          cout << "寫入資料錯誤" << endl;
93      else
94      {
```

```
95          if (!readData(rStruct))  // 讀取資料
96              cout << " 讀取資料錯誤 " << endl;
97          else
98              showData(rStruct);  // 顯示資料
99      }
100
101     system("pause");
102 }
```

程式講解

1. 程式碼第 1-3 行引入所需要的標頭檔與宣告使用 std 命名空間。第 5 行定義常數 MAX_NUM 等於 2，表示最多只能有 2 筆資料。
2. 程式碼第 7-12 行定義 BMI 基本資料的結構 _MYSTRUCT；其中包含了 3 個變數成員：字串變數 name、整數變數 age 與長度等於 2 的浮點數陣列 data，分別表示姓名、年齡與身高和體重；陣列 data 的第 1 個元素表示身高，第 2 個元素表示體重。
3. 程式碼第 15-26 行爲自訂函式 addData() 的程式本體，此函式用於輸入 MAX_NUM 筆的 BMI 基本資料；並帶有一個 _MYSTRUCT 陣列型別的參數 myStruct。第 17-25 行使用 for 重複敘述輸入 MAX_NUM 筆的 BMI 基本資料。由於結構陣列變數的傳遞是傳址呼叫，因此在此函式中所新增的結構陣列資料，也會直接改變原來呼叫者所傳入的結構陣列內容。
4. 程式碼第 29-46 行爲自訂函式 writeData() 的程式本體，用於將 MAX_NUM 筆的 BMI 基本資料寫入檔案；並帶有一個 _MYSTRUCT 陣列型別的參數 myStruct。第 32 行宣告布林變數 fg，並且初始值等於 true，預先表示建立檔案與寫入檔案資料並沒有發生錯誤。第 35-37 行使用開檔模式 ios::binary 與 ios::out 建立檔案 doc.data，所以此檔案爲供寫入資料的二進位檔案；若建立檔案失敗則將變數 fg 設定爲 false。

   程式碼第 40 行使用 write() 函式將包含 MAX_NUM 筆的 BMI 基本資料的結構陣列 myStruct 一次寫入檔案；因此，需要指定寫入資料的起始位址：(char *)&myStruct[0]，以及資料的寫入長度：sizeof(_MYSTRUCT)×MAX_NUM。

   第 41-42 行使用 good() 函式判斷資料是否寫入成功，若寫入失敗則將變數 fg 設定爲 false。程式碼第 43 行使用 close() 函式關閉檔案；第 45 行回傳變數 fg。
5. 程式碼第 48-70 行爲自訂函式 readData() 的程式本體，並帶有一個 _MYSTRUCT 陣列型別的參數 rStruct；此函式用於從檔案中讀取資料，並儲存到參數 rStruct。第 54-56 行使用開檔模式 ios::binary 與 ios::in 開啓檔案 doc.data，所以此檔案爲供讀取資料的二進位檔案；若開啓檔案失敗則將變數 fg 設定爲 false。

第 58-66 行示範使用 for 重複敘述每次只讀取一筆資料，因此會讀取 MAX_NUM 次的資料。第 60 行使用 read() 函式從檔案讀取資料，每次讀取的資料量爲 sizeof(_MYSTRUCT) 位元組。第 61-65 行使用 good() 函式判斷資料是否讀取成功，若讀取失敗則將變數 fg 設定爲 false 並離開 for 重複敘述。程式碼第 68-69 行使用 close() 函式關閉檔案並回傳變數 fg。

6. 程式碼第 73-83 行爲自訂函式 showData() 的程式本體，並帶有一個 _MYSTRUCT 陣列型別的參數 rStruct；此函式用於顯示 BMI 的基本資料。第 76-82 行使用 for 重複敘述逐一顯示結構陣列 rStruct 中的資料。
7. 程式碼第 87 行宣告 2 個 _MYSTRUCT 結構型別的陣列變數，陣列長度都等於 MAX_NUM。變數 myStruct 用於儲存使用者所輸入的資料，以及將資料寫入檔案。變數 rStruct 則使用於讀取檔案的資料。
8. 程式碼第 89 行呼叫自訂函式 addData() 新增資料，並將變數 myStruct 當成引數。第 91 行呼叫自訂函式 writeData() 將變數 myStruct 寫入檔案，若寫入資料成功則執行第 94-99 行。第 95 行呼叫自訂函式 readData() 從檔案讀取資料並儲存於變數 rStruct；若讀取成功則執行第 98 行，呼叫自訂函式 showData() 顯示變數 rStruct 裡的資料。

三、範例程式解說

1. 建立專案，程式碼第 1-4 行引入所需要的標頭檔與宣告使用 std 命名空間。

```
1 #include <iostream>
2 #include <fstream>
3 #include <conio.h>
4 using namespace std;
```

2. 程式碼第 6-12 行定義結構 _BOXEDMEAL，作爲訂購便當的基本資料。其中有 4 個變數成員：作爲姓名的字串變數 name、作爲電話號碼的字串變數 phone、作爲選擇葷食或是素食的布林變數 veg、以及作爲選擇排骨便當或是雞腿便當的布林變數 porkChop。

```
6  struct _BOXEDMEAL
7  {
8      char name[20];  // 姓名
9      char phone[20];  // 電話
10     bool veg; //true: 素食 false: 葷食
11     bool porkChop; //true: 排骨便當 false: 雞腿便當
12 };
```

3. 程式碼第 15-22 行爲自訂函式 showMenu() 的程式本體，此函式用於顯示功能選單。

```
15 void showMenu()
16 {
17     system("cls");
18     cout << "1. 新增資料 " << endl;
19     cout << "2. 顯示資料 " << endl;
20     cout << "3. 結束 " << endl;
21     cout << " 輸入選擇 (1-3): ";
22 }
```

4. 程式碼第 25-42 行爲自訂函式 writeData() 的程式本體，並帶有一個 _BOXEMEAL 結構型別的參數 meal；此函式用於將訂購便當的基本資料 meal 儲存於檔案。第 28 行宣告布林變數 fg，初始值等於 true，即預設成功地將資料寫入檔案。第 30-32 行使用 ios::binary|ios::out|ios::app 此 3 個開檔模式開啓檔案 doc.data；所以此檔案爲二進位檔案，並可以附加資料。

   若檔案開啓失敗則將變數 fg 設定爲 false；若檔案開啓成功則執行第 34-40 行。第 35-37 行使用 write() 函式將資料 meal 寫入檔案，並使用 good() 函式判斷若寫入資料失敗則將變數 fg 設定爲 false。第 39 行使用 close() 函式將檔案關閉。第 41 行回傳變數 fg，表示寫入資料成功或是失敗。

```
25 bool writeData(_BOXEDMEAL meal)
26 {
27     fstream file;
28     bool fg = true;
29
30     file.open("doc.data", ios::binary | ios::out|ios::app);
31     if (!file)
32         fg = false;
33     else
34     {
35         file.write((char*)&meal, sizeof(_BOXEDMEAL));
36         if (!file.good())
37             fg = false;
38
39         file.close();
40     }
41     return fg;
42 }
```

5. 程式碼第 45-59 行爲自訂函式 showData() 的程式本體，並帶有一個 _BOXEMEAL 結構型別的參數 meal；此函式用於顯示訂購便當的基本資料 meal。第 49-57 行判斷如果變數 meal.veg 等於 true 表示便當爲素食，否則爲葷食；若是葷食則進一步判斷 meal.porkChop 若等於 true 表示爲排骨便當，否則爲雞腿便當。

```
45 void showData(_BOXEDMEAL meal)
46 {
47     cout << "姓名：" << meal.name << endl;
48     cout << "年齡：" << meal.phone << endl;
49     if (meal.veg)
50         cout << "素食" << endl;
51     else
52     {
53         if (meal.porkChop)
54             cout << "豬排便當" << endl;
55         else
56             cout << "雞腿便當" << endl;
57     }
58     cout << endl;
59 }
```

6. 程式碼第 62-81 行爲自訂函式 readData() 的程式本體，此函式用於從檔案中讀取訂購便當的資料，並呼叫自訂函式 showData() 顯示這些資料。第 65 行宣告 _BOXEDMEAL 結構型別的變數 meal，用於儲存從檔案讀取的資料。第 66 行宣告布林變數 fg，初始值等於 true，即預設成功地從檔案讀取資料。第 68 行使用開檔模式 ios::binary|ios::in 開啓檔案 doc.data；所以此檔案爲二進位檔案，並用於讀取資料。

   第 69-70 行若檔案開啓失敗則將變數 fg 設定爲 false，若檔案開啓成功，則執行第 72-78 行。第 74 行使用 while() 重複敘述持續從檔案讀取資料；並且將讀取資料的 read() 函式直接當作爲 while() 的執行條件。第 75 行呼叫自訂函式 showData() 顯示所讀取的資料 meal。當所有資料都讀取並顯示之後，第 77 行呼叫 close() 函式關閉檔案。第 80 行回傳變數 fg，表示讀取資料成功或是失敗。

```
62 bool readData()
63 {
64     fstream file;
65     _BOXEDMEAL meal;
66     bool fg = true;
67
```

```
68      file.open("doc.data", ios::binary | ios::in);
69      if (!file)
70          fg = false;
71      else
72      {
73          cout << "----- 顯示資料 -----" << endl;
74          while (file.read((char*)&meal, sizeof(_BOXEDMEAL)))
75              showData(meal);
76
77          file.close();
78      }
79
80      return fg;
81  }
```

7. 程式碼第 84-114 行爲自訂函式 addData() 的程式本體，用於新增訂購便當的基本資料。第 86 行宣告 _BOXEDMEAL 結構型別的變數 meal，用於儲存使用者所輸入的資料。第 88-113 爲 while 無窮迴圈，第 90-91 行顯示提示訊息與讀取使用者所輸入的姓名，並儲存於結構 meal 的資料成員 name，第 92-93 行使用 strcmp() 函式判斷 meal.name 的內容若等於 "-1" 則表示結束輸入資料。

   第 95-98 行讀取使用者所輸入的電話與選擇葷食還是素食。第 100-104 行判斷若選擇了葷食，則第 102-103 行需要再輸入選擇哪種的葷食便當。第 106 行呼叫 writeData() 自訂函式，並將訂餐資料 meal 作爲引數。若資料寫入檔案失敗，則第 108-109 行顯示錯誤訊息並返回呼叫者，否則第 112 行顯示資料新增成功的訊息。

```
84  void  addData()
85  {
86      _BOXEDMEAL meal;
87
88      while (true)
89      {
90          cout << "\n 輸入姓名 ( 輸入 -1 結束輸入資料 )：";
91          cin >> meal.name;
92          if (strcmp(meal.name, "-1")==0)
93              return;
94
95          cout << " 輸入電話：";
96          cin >> meal.phone;
97          cout << "(0) 葷食 (1) 素食：";
98          cin >> meal.veg;
```

```
 99
100         if (!meal.veg)
101         {
102             cout << "(0) 雞腿便當 (1) 排骨便當：";
103             cin >> meal.porkChop;
104         }
105
106         if (!writeData(meal))
107         {
108             cout << " 新增資料失敗 " << endl;
109             return;
110         }
111         else
112             cout << " 新增資料成功 " << endl;
113     }
114 }
```

8. 開始於 main() 主函式中撰寫程式。程式碼第 118 行宣告整數變數 sel，用於儲存使用者所輸入的功能編號。第 120-141 行爲 while 無窮迴圈，第 122 行呼叫自訂函式 showMenu() 顯示功能選單，第 123 行讀取使用者所輸入的功能編號，並儲存於變數 sel。第 125-138 行爲 switch…case 選擇敘述，並根據變數 sel 執行相對應的功能。第 127-128 爲「新增資料」的功能，呼叫自訂函式 addData()。第 130-131 爲「顯示資料」的功能，呼叫自訂函式 readData()。程式碼第 133-134 行爲「結束」的功能，呼叫函式 exit() 結束程式。第 136-137 行爲 default 區塊；若輸入錯誤的選項則顯示錯誤訊息。第 139-140 行等待使用者按任一鍵後繼續執行。

```
118 int sel;
119
120 while (true)
121 {
122     showMenu();
123     cin >> sel;
124
125     switch (sel)
126     {
127         case 1: addData();
128             break;
129
130         case 2: readData();
131             break;
```

```
132
133          case 3: exit(0);
134              break;
135
136          default: cout << " 輸入錯誤，";
137              break;
138      }
139      cout << " 按鍵繼續 ...";
140      while (!_kbhit());
141  }
142
143  system("pause");
```

重點整理

1. 文字檔可以直接使用一般的文字軟體開啓，也能直接看得懂內容，方便容易編輯與了解。而無法以文字描述的資料則適合使用二進位檔案的形式儲存。例如影像所儲存的是像素點的色彩值。因此，並沒有哪一種檔案形式會比較好，而是各有不同的使用時機。

分析與討論

1. 讀寫檔案所使用的函式也都能使用於二進位檔案；但有的函式在寫入資料時仍然會以文字的方式處理；例如：對於 "<<" 運算子而言，寫入檔案的資料都是以文字的方式儲存。所以對於二進位檔案而言，通常都是使用 `read()` 與 `write()` 函式才能達到寫入二進位資料的效果。然而，對於字元、字串仍然是以文字的方式寫入檔案。此外，換行字元寫入二進位檔案，其值爲 `0A` 而並非 `0A0D`。
2. 本範例第 25-1 節示範了如何把一維陣列的資料儲存到二進位檔案，以及如何從二進位檔案中讀取資料並儲存到一維陣列；對於多維陣列資料的存取仍然是使用相同的方式。例如：有一個 `int` 型別的 2×3 的二維陣列 `arr`，將此陣列寫入二進位檔案 `file` 的方式爲：

```
1  int arr[2][3] = { {1,2,3},{4,5,6} };
2
3  file.write((char*)&arr, sizeof(arr));
```

從檔案讀取此資料，並儲存到二維陣列的方式爲：

```
1  int arr[2][3];
2
3  file.read((char*)&arr, sizeof(arr));
```

3. 本範例第 25-1 節在示範將 string 型別的資料轉爲字元陣列型別的字串時，需要宣告空間足夠的字元陣列，用以儲存轉換後的字串資料。然而，程式在執行過程中，無法知道 string 型別的字串變數經過輸入、修改之後，在儲存到檔案之前其長度是多少；因此，往往需要宣告一個空間挺大的字元陣列，才不至於擔心 string 型別的字串資料傳換爲字元陣列字串之後，此字元陣列的空間無法容納這些轉換後的資料。

   所以這樣的做法會浪費不少空間，此字元陣列儲存至檔案時也同時存進去了許多沒有意義的資料（因爲字元陣列的空間比字串資料還要長）。因此，可以改使用動態記憶體配置的方法，只配置恰好儲存 string 型別的字串所需要的空間；如此便不需要事先宣告一個很大的字元陣列。

   如下範例程式所示，假設二進位檔案 file 已經開啓，程式碼第 7 行計算字串 str1 的長度，第 8 行使用 new 配置長度等於 len 的空間給字元指標 ptr1。第 10 行使用 sprintf() 函式將字串變數 str1 轉換爲字元陣列的字串內容並儲存於指標變數 ptr1。第 11 行使用 write() 函式將指標 ptr1 所指的內容寫入檔案。第 13 行釋放指標 ptr1 所佔有空間。

```
 1 #pragma warning(disable : 4996)
 2
 3 string str1 = " 王小明 ";
 4 char* ptr1,*ptr2;
 5 int len; // 字串長度
 6
 7 len = str1.length()+1;  // 加 1 爲字串結尾字元 '\0' 的長度
 8 ptr1 = new char[len];
 9
10 sprintf(ptr1, "%s", str1.c_str());
11 file.write(ptr1, len);
12 file.close();
13 delete []ptr1;
```

   從檔案中讀取資料的方式也相同，如下所示。第 14 行使用 new 配置長度等於 len 的空間給字元指標 ptr2。第 15 行使用 read() 函式從檔案讀取資料並儲存到指標 ptr2 所指的位址空間。第 18 行釋放所配置的記憶體空間。

```
14 ptr2 = new char[len];
15 file.read(ptr2, len);
16 file.close();
17 cout << ptr2 << endl;
18 delete []ptr2;
```

程式碼列表

```
#include <iostream>
#include <fstream>
#include <conio.h>
using namespace std;

struct _BOXEDMEAL
{
    char name[20];  // 姓名
    char phone[20];  // 電話
    bool veg; //true: 素食 false: 葷食
    bool porkChop; //true: 排骨便當 false: 雞腿便當
};

//-------- 顯示選單 -----------------
void showMenu()
{
    system("cls");
    cout << "1. 新增資料 " << endl;
    cout << "2. 顯示資料 " << endl;
    cout << "3. 結束 " << endl;
    cout << " 輸入選擇 (1-3): ";
}

//------------- 寫入資料 --------------
bool writeData(_BOXEDMEAL meal)
{
    fstream file;
    bool fg = true;

    file.open("doc.data", ios::binary | ios::out|ios::app);
    if (!file)
        fg = false;
    else
    {
        file.write((char*)&meal, sizeof(_BOXEDMEAL));
        if (!file.good())
            fg = false;

        file.close();
    }
    return fg;
```

```
42 }
43
44 //------------- 顯示資料 ----------------
45 void showData(_BOXEDMEAL meal)
46 {
47     cout << "姓名：" << meal.name << endl;
58     cout << "年齡：" << meal.phone << endl;
59     if (meal.veg)
50         cout << "素食" << endl;
51     else
52     {
53         if (meal.porkChop)
54             cout << "豬排便當" << endl;
55         else
56             cout << "雞腿便當" << endl;
57     }
58     cout << endl;
59 }
60
61 //------------- 讀取資料 ----------------------------
62 bool readData()
63 {
64     fstream file;
65     _BOXEDMEAL meal;
66     bool fg = true;
67
68     file.open("doc.data", ios::binary | ios::in);
69     if (!file)
70         fg = false;
71     else
72     {
73         cout << "----- 顯示資料 -----" << endl;
74         while (file.read((char*)&meal, sizeof(_BOXEDMEAL)))
75             showData(meal);
76
77         file.close();
78     }
79
80     return fg;
81 }
82
```

```
//------------- 填寫資料 ------------------
void  addData()
{
    _BOXEDMEAL meal;

    while (true)
    {
        cout << "\n 輸入姓名 ( 輸入 -1 結束輸入資料 )：";
        cin >> meal.name;
        if (strcmp(meal.name, "-1")==0)
            return;

        cout << " 輸入電話：";
        cin >> meal.phone;
        cout << "(0) 葷食 (1) 素食：";
        cin >> meal.veg;

        if (!meal.veg)
        {
            cout << "(0) 雞腿便當 (1) 排骨便當：";
            cin >> meal.porkChop;
        }

        if (!writeData(meal))
        {
            cout << " 新增資料失敗 " << endl;
            return;
        }
        else
            cout << " 新增資料成功 " << endl;
    }
}

int main()
{
    int sel;

    while (true)
    {
        showMenu();
        cin >> sel;

```

```
125          switch (sel)
126          {
127              case 1: addData();
128                  break;
129
130              case 2: readData();
131                  break;
132
133              case 3: exit(0);
134                  break;
135
136              default: cout << " 輸入錯誤，";
137                  break;
138          }
139          cout << " 按鍵繼續 ...";
140          while (!_kbhit());
141      }
142
143      system("pause");
144  }
```

本章習題

1. 寫一程式，每執行一次程式就會產生 5 個介於 0-100 的亂數，以附加的方式寫入二進位檔案。
2. 寫一程式，讀取由第 1 題所建立的檔案內容。
3. 寫一程式讓使用者輸入 5 次的字串，輸入結束之後建立二進位檔案，再將此 5 個字串寫入檔案。字串需使用字元陣列。
4. 寫一程式，讀取由第 3 題所建立的檔案內容。
5. 改修第 3 題，每執行一次程式便會在檔案增加 5 個字串。
6. 寫一程式，讀取由第 5 題所建立的檔案內容。
7. 寫一程式，輸入 3 種行動電話的基本資料並寫入二進位檔案。基本資料包含：行動電話的型號與價錢，並以結構表示。
8. 寫一程式，讀取由第 7 題所建立的檔案內容。