Example 24

檔案處理：文字檔

一共有 5 位學生，每位學生有 2 項基本資料：姓名與年齡，並使用結構表示學生資料。寫一程式，使用者可以新增資料或是顯示資料。新增一筆資料之後立即將資料儲存於檔案，顯示資料時會立即從檔案讀取並顯示資料。

一、學習目標

程式執行過程中會產生資料，例如：從網站下載的資料、使用文書軟體編寫了文件、使用繪圖軟體畫了一張畫；這些都可以稱爲資料。這些資料必須以檔案的形式儲存於永久儲存媒體，才不至於電腦關機後，這些資料便消失；而檔案仍然可以再次載入繼續編輯。

C++ 將處理檔案視爲對串流的處理，因此簡化了存取檔案的細節操作。依照檔案資料所儲存的方式，大致上可以分爲文字檔案與二進位檔案。若以存取的方式，還可以分爲循序存取與隨機存取。這些分類方式都是因應實際運用之需要而產生的格式。

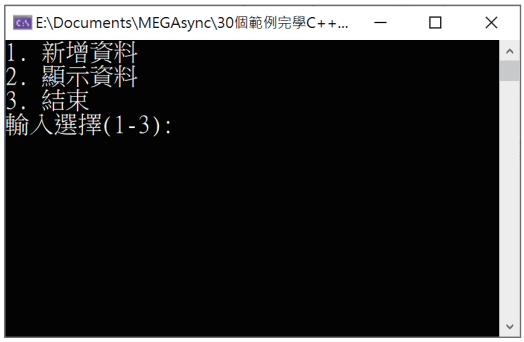
文字檔案可以使用一般的文字處理軟體開啓與編修；因此便利與簡單是文字檔案的優點。而無法以文字形式儲存的檔案，便是二進位檔案，例如：程式執行檔、影像、音樂等。

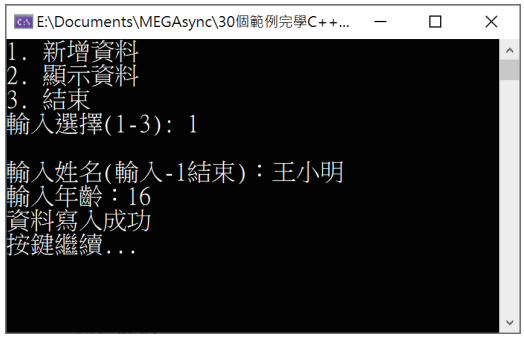
檔案開啓後須按照順序讀取或寫入資料，便稱之爲循序存取。而隨機存取則可以指定要讀取或是寫入資料的位置。

本範例首先了解檔案的處理流程，接著練習如何建立、寫入與讀取文字檔案；最後再學習如何將結構寫入檔案，以及如何從檔案中讀取結構資料。

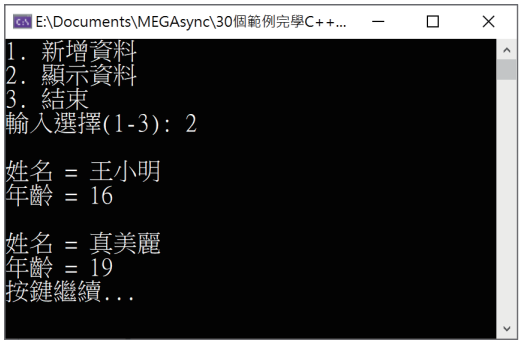
二、執行結果

下圖左爲原始畫面，有 2 項主要的功能：「新增資料」與「顯示資料」。程式開始執行時會先讀取先前的資料，如此才能知道已經有多少位的學生資料。選擇「新增資料」後輸入姓名與年齡，並儲存資料；如下圖右所示。





當輸入的資料已經到達最大的人數，則無法再輸入資料，如下圖左所示。選擇「顯示資料」可以顯示目前已經輸入的學生資料，如下圖右所示。若程式結束之後再開啓程式，一樣可以繼續新增學生資料。



24-1　檔案處理流程

無論是讀取檔案中的資料或將資料寫入檔案，都有一定的步驟與流程；而不同組成方式的檔案也有一定的使用方式。

檔案與串流

程式執行過程中會產生資料，例如：從網路下載的音樂、股票預測的計算結果、遊戲的分數與進度、繪製到一半的影像等；這些資料隨著程式結束而消失；因此，必須要將這些資料儲存於諸如硬碟、USB 隨身碟等永久儲存媒體，這些儲存的資料便稱之爲檔案。當下次執行程式時，可以載入這些檔案繼續未完成的工作。

將資料讀入到程式中，稱爲輸入（Input）；這些資料的來源可能來自檔案、鍵盤、網路、攝影機、光碟等。將程式中的資料儲存於檔案、網路硬碟、送往印表機列印、上傳至某個雲端等，稱爲輸出（Output）。

然而，無論這些資料來自於何處或要往何處去，都在彼此交流互動。不同的輸出入（I/O）設備或來源都有不同的特性與特定的存取方式；若要爲特定的 I/O 來源或設備撰寫程式與操作方式，便顯得麻煩與困難。

因此，C++ 將這些不同 I/O 來源或裝置的資料，都視爲串流（Stream，或資料流）；即如同一條河流上有來自於各處的資料，這些資料也會往不同的地方流去。程式設計師不需要去理會這些資料來源處的特性，或資料要送去的地方的獨特處理方式；只需使用 C++ 處理串流所提供的類別與函式，把資料送出去或儲存起來便可以了。如此一來，便可以簡化檔案在處理時的麻煩與減少錯誤。

檔案開啓模式

依據對檔案不同的操作形式：讀取資料、寫入資料、附加資料等不同的操作方式，開啓檔案時可有以下的開檔模式（開啓模式）；這些開檔模式位於 `std` 命名空間的 `ios_base` 類別。

常數	說明
`app`	附加模式。將資料索引移至檔尾，因此新增的資料會附加於檔。檔案若不存在，會產生新檔案。
`ate`	檔案開啓後，將檔案索引移至檔尾。
`binary`	二進位檔案模式。
`in`	讀取模式，檔案供讀取資料。檔案必須先存在才能開啓檔案。
`out`	寫入模式，檔案供寫入資料。檔案若不存在，會產生新的檔案。
`trunc`	檔案若已存在，則清空檔案內容。檔案若不存在，會產生新的檔案。

這些常數可以使用 `"|"` 運算子組合，例如：開啓供寫入資料的檔案，並且將新增的資料附加於檔尾。

```
ios_base::out | ios_base::app
```

或者使用 `ios` 命名空間也可以：

```
ios::out | ios::app
```

開啓與關閉檔案

有 3 種用於宣告檔案（串流）變數的資料類別：`fstream` 類別、`ifstream` 類別與 `ofstream` 類別；此 3 個類別都需要引入 `fstream` 標頭檔。`fstream` 爲 file stream 此 2 個字的縮寫。`ifstream` 爲 input file stream 此 3 個字的縮寫。而 `ofstream` 則爲 output file stream 此 3 個字的縮寫。此 3 個類別的寬字原版本分別爲：`wfstream`、`wifstream` 與 `wofstream`。

fstream 類別提供可以讀取或寫入的檔案，ifstream 類別專門提供做爲讀取資料的檔案，而 ofstream 類別則提供寫入資料的檔案。此 3 種類別所宣告的檔案，都使用 open() 函式開啓檔案。當檔案不再使用時則使用 close() 關閉檔案。以 fstream 類別所宣告的檔案變數爲例，如下所示：

```
1 fstream file;                        // 使用 fstream 類別宣告檔案變數
2
3 file.open("doc.txt", ios::in); // 使用函式 open() 開啓檔案
4         ⋮
5 file.close();                        // 使用函式 close() 關閉檔案
```

程式碼第 1 行使用 fstream 類別宣告檔案變數 file。第 3 行使用函式 open() 在目前的工作路徑開啓檔案 doc.txt，並且開檔模式爲 ios::in，所以此檔案用於讀取資料。當對檔案內的資料處理結束之後，第 5 行最後使用函式 close() 關閉檔案。

判斷檔案是否成功開啓

使用函式 open() 開啓檔案有可能會失敗；因此，當檔案開啓之後可使用 is_open() 函式判斷檔案是否開啓成功。檔案開啓成功則 is_open() 函式回傳 true，否則回傳 false。如下所示：

```
1 fstream file;
2
3 file.open("doc.txt", ios::in);
4 if (file.is_open() != true)    // 使用 is_open() 函式判斷檔案是否開啓成功
5 {
6     開啓檔案失敗的處理 ;
7 }
8     ⋮
9 file.close();
```

還有另一種更簡便的判斷方式，如下所示；程式碼第 4 行的 "!file" 表示檔案開啓失敗。

```
4 if (!file)    // 檔案開啓失敗
5 {
6     開啓檔案失敗的處理 ;
7 }
8     ⋮
9 file.close();
```

練習 1：建立新檔案

在當前的工作路徑之下，建立一個新的檔案 doc.txt，作爲儲存資料使用；並判斷是否成功建立檔案。

解說

因爲要建立新的檔案，所以可以使用 `fstream` 或是 `ofstream` 類別，並且開檔模式需要使用 `ios::out`。若還需要考量會後續增加資料，因此開檔模式還額外需要 `ios::app`。若是要建立可供同時寫入與讀取的檔案，則開檔模式使用 `ios::out|ios::in`。

建立檔案之後，再使用 `system("dir")` 顯示當前路徑下的所有檔案，便可以看到剛才建立的檔案 doc.txt。

執行結果

```
檔案建立成功，按鍵繼續。
磁碟區 E 中的磁碟是 E
磁碟區序號：  CA41-02C6

E:\Tmp\chapter 24\practice\01 的目錄

2020/10/11  上午 10:33    <DIR>          .
2020/10/11  上午 10:33    <DIR>          ..
2020/10/11  上午 10:31           107,520 ConsoleApplication1.exe
2020/10/11  上午 10:33                 0 doc.txt ◄── 剛建立的檔案
2020/10/04  下午 03:13            67,072 Say.exe
               3 個檔案         174,592 位元組
               2 個目錄  85,816,291,328 位元組可用
```

程式碼列表

```
 1  #include <iostream>
 2  #include <fstream>
 3  #include <conio.h>
 4  using namespace std;
 5
 6  int main()
 7  {
 8      ofstream outfile;
 9
10      outfile.open("doc.txt", ios::out | ios::app);
```

```
11      if (outfile.is_open())
12      {
13          cout << " 檔案建立成功，按鍵繼續。" << endl;
14          system("dir");
15          while (!_kbhit());
16          outfile.close();
17      }
18      else
19          cout << " 檔案建立失敗。";
20  }
```

程式講解

1. 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。
2. 程式碼第 8 行使用 ofstream 類別宣告檔案變數 outfile。
3. 程式碼第 10 行使用 open() 函式開啓檔案並使用開檔模式 ios::out|ios::app；所以會在當前的工作路徑之下，建立檔案 doc.txt。
4. 程式碼第 11 行使用 is_open() 函式判斷若檔案開啓成功，則執行程式碼第 12-17 行。
5. 程式碼第 13 行顯示檔案開啓成功的訊息，第 14 行使用函式 system() 與指令 "dir" 顯示當前路徑下的檔案與目錄。
6. 程式碼第 15-16 行讀取任意按鍵之後，使用函式 close() 關閉檔案。

24-2 將資料寫入檔案

建立新的檔案，可以使用 fstream 或是 ofstream 類別來建立檔案變數；並搭配 ios::out 開檔模式。需特別注意，使用 ios:out 開檔模式開啓已存在的檔案，此檔案的內容會被清除，視同使用 ios::out|ios::trunc。若要保留檔案原先的內容，或是繼續增加資料；則需使用 ios::out|iops::app 開檔模式。

若使用的開檔模式爲 ios::out|ios::in，則此檔案可以寫入與讀取資料，並且不會在檔案開啓時清空原有的內容；然而，一個檔案同時提供寫入資料與讀取資料，需要控制好讀取檔案的索引位置。須注意一點：檔案必須先建立之後，才能使用 ios::out|ios::in 開檔模式；否則在還沒有建立檔案的時候是無法使用 ios::out|ois::in 中的 ios::in 的模式，也因此會發生錯誤。

有 2 種方式將資料寫入檔案：使用 "<<" 運算子、以及使用 fstream 或 ofstream 類別所提供的函式。

使用 "<<" 運算子將資料寫入檔案

使用 "<<" 運算子將資料寫入檔案是最簡便的方式；例如以下範例：

```
1  #include <iostream>
2  #include <fstream>                    // 引入 fstream 標頭檔
3  using namespace std;
4
5  int main()
6  {
7      fstream file;                      // 使用 fstream 宣告檔案變數
8
9      file.open("doc.txt", ios::out );   // 使用函式 open() 開啓檔案
10     file << " 使用 fstream 寫入資料 ";   // 使用 "<<" 運算子將資料寫入檔案
11     file << "Hello\n" << 10 << endl;   // 再使用 "<<" 運算子寫入資料
12     file.close();                      // 最後用函式 close() 關閉檔案
13 }
```

程式碼第 2 行引入 fstream 標頭檔；因此，在第 7 行才可以使用 fstream 類別宣告檔案變數 file。第 9 行使用 open() 函式在目前的工作路徑建立檔案 doc.txt，並且開檔模式爲 ios::out，所以此檔案供寫入資料。第 10 行使用 "<<" 運算子將字串 " 使用 fstream 寫入資料 " 寫入檔案。第 11 行再次使用 "<<" 運算子將字串 "Hello\n" 與數字 10 寫入檔案。第 12 行使用函式 close() 關閉檔案。

練習 2：使用 "<<" 運算子寫入資料

輸入姓名與分數後再儲存於檔案 doc.txt；並判斷是否資料是否成功儲存於檔案。

解說

因爲是要將輸入的姓名與分數寫入檔案，所以可以使用 fstream 或是 ofstream 類別，開檔模式需要使用 ios::out；並使用 "<<" 運算子將姓名與分數寫入檔案即可。要判斷是否成功將資料寫入檔案，則使用函式 is_fail()（請參考分析與討論的第 2 點）。

因爲建立的檔案是文字檔；因此，即使欲儲存的資料是數值或是數值變數，也是以文字的形式儲存到檔案。

執行結果

儲存成功之後，可以至目前的工作路徑下，以文字處理軟體開啓 doc.txt；便可以看到檔案內有 2 列資料：王小明、90。

```
輸入姓名：王小明
輸入分數：90
資料已儲存
```

程式碼列表

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ofstream outfile;
    string name;
    string score;

    cout << " 輸入姓名：";
    cin >> name;
    cout << " 輸入分數：";
    cin >> score;

    if (name == "-1")
        exit(0);

    outfile.open("doc.txt", ios::out);
    if (outfile.is_open())
    {
        outfile << name << endl;
        outfile << score << endl;
        if(!outfile.fail())
           cout << " 資料已儲存 " << endl;
        outfile.close();
    }
    else
        cout << " 無法建立檔案 " << endl;

    system("pause");
}
```

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。
2. 程式碼第 7-9 宣告檔案變數 outfile、姓名 name 與分數 score；分別用於建立檔案、儲存使用者輸入的姓名與分數。
3. 程式碼第 11-14 行輸入姓名與分數，並分別儲存於變數 name 與 score。
4. 程式碼第 16-17 行判斷若輸入的姓名等於 "-1"，則結束程式。
5. 程式碼第 19 行使用 fstream 類別的函式 open() 建立檔案 doc.txt，並且開檔模式爲 ios::out；因此，此檔案供寫入資料。
6. 程式碼第 20-29 行爲 if…else 判斷敘述，若檔案 doc.txt 建立成功則執行第 21-27 行。第 22-23 行使用 "<<" 運算子將變數 name 與 score 寫入檔案 doc.txt。第 24-25 行使用 fail() 函式判斷資料是否成功寫入檔案。26 行使用函式 close() 關閉檔案。

使用 fstream/ofstream 的函式寫入檔案

一共有 3 個與寫入串流相關的函式，如下表所示。

函式	說明
flush()	將在緩衝區內的資料寫入檔案。
put(a)	將字元 a 寫入檔案。a 為字元型別。
write(a,b)	將長度等於 b 的資料 a 寫入檔案。a 為 const char * 型別，b 為整數型別。

當資料寫入檔案之時，其實並不是馬上寫入檔案中；而是先存放於緩衝區（電腦的記憶體），等到作業系統有空、緩衝區已滿、程式結束或呼叫函式 close() 時，才眞正將緩衝區內的資料存放於檔案。然而，若是在尚未把緩衝區內的資料眞正寫入檔案時，恰巧程式發生了錯誤、作業系統出問題、電腦當機等；則資料將會遺失。因此，爲了避免這些問題發，可以在程式中呼叫 flush() 函式自行將緩衝區內的資料寫入檔案。

函式 put() 可以將 1 個字元寫入檔案。函式 write() 則可以將一塊記憶體的內容（例如：字串、陣列、或是指標所指的記憶體空間）寫入檔案。

如下範例，程式碼第 1 行宣告 fstream 類別的檔案變數 file。第 2 行宣告字串變數 name，內容等於 "Mary"。第 3 行宣告字元陣列形式的字串，內容等於 "Hello"。第 5 行使用 open() 函式開啓檔案 doc.txt，並且開檔模式爲 ios::out；因此，此檔案用於寫入資料。

```
1  fstream file;
2  string name = "Mary";
3  char arr[6] = "Hello";
4
5  file.open("doc.txt", ios::out);
6
7  file.put('C');
8  file.write("Spring",6);
9  file.write(name.c_str(), 3);
10 file.write((const char*)arr,strlen(arr));
11
12 file.close()
```

第 7 行使用 put() 函式將字元 'C' 寫入檔案，第 8 行使用 write() 函式將字串 "Spring" 的前 6 個字元寫入檔案；因此會將整個字串 "Spring" 寫入檔案。第 9 行將字串 name 使用 c_str() 函式轉型爲字元陣列形式的字串，並只取前 3 個字元寫入檔案："Mar"。第 10 行先將字元型別的一維陣列，轉型爲 const char* 型別之後再寫入檔案。第 12 行使用 close() 函式關閉檔案 file。

練習 3：函式 put() 與 write()

寫一程式可持續輸入姓名與分數，並儲存於檔案 doc.txt；姓名輸入 -1 則結束程式。重新執行程式後，可繼續新增資料。

解說

練習 3 與練習 2 的差別，在於可以不斷地輸入姓名與分數；以及重新執行程式時，原本儲存在檔案裡的資料不會被清空，後續輸入的資料會新增在檔案尾端。因此，開檔模式應爲：ios::out|ios::app。

執行結果

如下所示，可以連續輸入資料；當姓名輸入 "-1" 則結束程式。

```
輸入姓名：王小明
輸入分數：90
輸入姓名：真美麗
輸入分數：85
輸入姓名：李小強
輸入分數：89
輸入姓名：-1
```

程式碼列表

```
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    ofstream outfile;
    string name;
    int score;
    char arr[5];

    outfile.open("doc.txt", ios::out|ios::app);
    if (!outfile.is_open())
    {
        cout << "無法開啓檔案" << endl;
        exit(0);
    }

    while (true)
    {
        cout << "輸入姓名：";
        cin >> name;
        if(name == "-1")
            break;

        cout << "輸入分數：";
        cin >> score;

        outfile.write(name.c_str(),name.length());
        outfile.put('\n');

        _itoa_s(score, arr, 10);
        outfile.write(arr,strlen(arr));
        outfile.put('\n');
    }

    outfile.close();
    system("pause");
}
```

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。
2. 程式碼第 7 行宣告 ofstream 類別的檔案變數 outfile。第 8-9 行宣告字串變數 name 與整數變數 score，用於儲存輸入的姓名與分數。第 10 行宣告字元型別的一維陣列，用於儲存分數轉換爲字串後的儲存空間。
3. 程式碼第 12 行使用 open() 開啓檔案 doc.txt，開檔模式爲 ios::out|ios::app；因此，此檔案用於寫入資料，並且可以將資料附加於檔案尾端。第 13-17 行使用 is_open() 函式判斷若檔案開啓失敗，則顯示錯誤訊息並結束程式。
4. 程式碼第 19-35 行爲 while 無窮迴圈。第 21-24 行輸入姓名並儲存於變數 name，若變數 name 的內容等於 "-1"，則結束程式。第 26-27 行輸入分數並儲存於變數 score。
5. 第 29-30 行使用 write() 函式將變數 name 寫入檔案；第 30 行使用 put() 函式把換行字元 '\n' 寫入檔案，如此才不會和下一筆資料寫入檔案的相同一列位置。
6. 程式碼第 32 行先使用 _itia_s() 函式將變數 score 轉換爲字串，並儲存於字元陣列 arr。第 33 行再使用 write() 函式將變數 arr 寫入檔案。第 34 行使用 put() 函式把換行字元 '\n' 寫入檔案。
7. 當不再輸入資料，離開 while 無窮迴圈之後，程式碼第 37 行使用 close() 關閉檔案。

24-3　從檔案讀取資料

從檔案讀取資料可以使用 fstream 或 ifstream 類別，開檔模式則使用 ios::in。若檔案不存在則會發生錯誤；因此，開啓檔案之後通常會先檢查檔案是否成功開啓。讀取檔案內的資料有 2 種方式：使用 ">>" 運算子、以及使用 fstream 或 ifstream 類別所提供的函式。

使用 ">>" 運算子讀取檔案

使用 >> 運算子讀取檔案資料是最簡便的方式；如下範例所示。第 11 行使用 ios::in 開檔模式，第 12 行使用 ">>" 運算子從檔案 file 讀取一列資料，並儲存於變數 str。第 13 行使用 ">>" 運算子從檔案 file 讀取一列資料，並儲存於變數 num。

```
 6     ⋮
 7  ifstream file;
 8  string str;
 9  int num
10
11  file.open("doc.txt", ios::in );  // 使用 ios::in 開檔模式讀取檔案資料
12  file >> str;                     // 使用 ">>" 運算子從檔案讀取資料
13  file >> num;
14  file.close();
```

此處須注意，雖然從文字檔案讀取的資料都是文字（字串），但若讀取數值型的文字並儲存到數值型別的變數時，">>" 運算子會自動做適當的轉型；如第 13 行所示。此外，如果原資料有空白字元，則使用 ">>" 讀取資料時，空白字元會被略過；例如：將字串 "abc　　" 寫入檔案，則讀取資料時只會讀取到 "abc"。

練習 4：使用 ">>" 運算子讀取資料

寫一程式，將姓名 " 王小明 " 與分數 90 寫入檔案 doc.txt 之後，再開啓檔案並讀取這 2 筆資料。

解說

此練習要開啓已經存在的檔案；因此，使用 ifstream 類別來宣告檔案變數，並且開檔模式爲 ios::in。在檔案內只有 2 筆形式簡單的資料，所以使用 ">>" 運算子來讀取檔案內的資料即可。

執行結果

如下所示，從檔案讀取資料：姓名與年齡，並顯示所讀取的資料。

```
姓名 = 王小明
分數 = 90
```

程式碼列表

```
 1 #include <iostream>
 2 #include <fstream>
 3 using namespace std;
 4
 5 bool writeData()
 6 {
 7     fstream file;
 8
 9     file.open("doc.txt", ios::out);
10     if (!file.is_open())
11     {
12         cout << " 無法建立檔案 " << endl;
13         return false;
14     }
15
16     file << " 王小明       " << endl;
```

```
17      file << 90 << endl;
18      file.close();
19      return true;
20  }
21
22  int main()
23  {
24      ifstream file;
25      string name;
26      int score;
27
28      if (!writeData())
29          exit(0);
30
31      file.open("doc.txt", ios::in);
32      if(!file.is_open())
33      {
34          cout << " 無法開啓檔案 " << endl;
35          exit(0);
36      }
37
38      file >> name;
39      file >> score;
40      file.close();
41
42      cout << " 姓名 = " << name << endl;
43      cout << " 分數 = " << score << endl;
44
45      system("pause");
46  }
```

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。
2. 程式碼第 5-20 行爲自訂函式 `writeData()`，此函式用於在目前的工作路徑下建立檔案 `doc.txt`，並寫入資料：" 王小明　　" 與 `90`；資料寫入檔案成功則回傳 `true`。第 16 行故意在 " 王小明 " 後面加上多個空白字元，雖然這些空白字元也會被寫入檔案，但是第 38 行讀取資料時，這些空白字元會被省略不計。
3. 程式碼第 24 行宣告 `ifstream` 類別的檔案變數 `file`，第 25-26 行宣告字串變數 `name` 與整數變數 `score`，分別用於儲存姓名與分數。

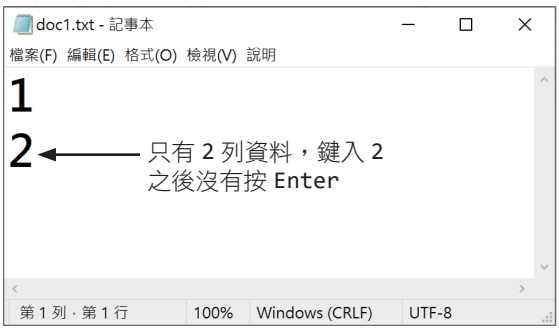
4. 程式碼第 28-29 行呼叫自訂函式 `writeData()` 建立檔案 `doc.txt`，並寫入資料；若回傳值等於 `false` 表示建立檔案失敗，則結束程式。
5. 程式碼第 31 行使用 `open()` 函式開啓檔案 `doc.txt`，開檔模式爲 `ios::in`；因此，此檔案只供讀取資料。第 32-36 行使用 `is_open()` 函式判斷開啓檔案是否成功，若開啓檔案失敗顯示錯誤訊並結束程式。
6. 程式碼第 38-39 行從檔案 `file` 讀取資料，並各自儲存於變數 `name` 與 `score`。第 40 行使用 `close()` 函式關閉檔案。雖然在第 16 行寫入資料時，故意在 " 王小明 " 之後留下許多的空白字元：" 王小明　　"，但第 38 行讀取時會略過這些空白字元，因此變數 `name` 只會等於 " 王小明 "。
7. 程式碼第 42-43 行顯示從檔案讀取的資料。

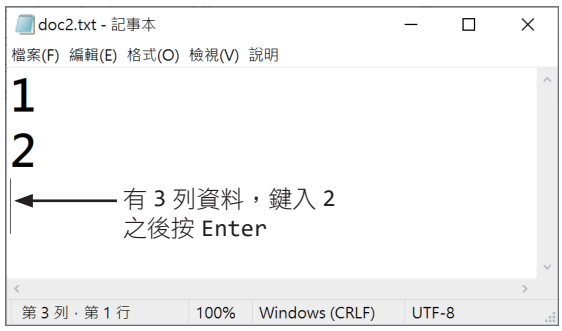
使用 eof() 函式判斷檔案結束

讀取檔案資料時，可以使用 `eof()` 函式判斷是否已到了檔案資料的尾端，無法再讀取任何資料；此時 `eof()` 回傳 `true`，表示已到了檔案尾端。例如：從檔案 `infile` 持續讀取資料，並儲存到變數 `str`，如下程式所示：

```
1          ⋮
2  while( !infile.eof())  // 判斷是否已經到了檔案尾端
3  {
4          ⋮
5      infile >> str;     // 從檔案讀取資料，並儲存於變數 str
6          ⋮
7  }
```

然而，使用 `eof()` 函式需要特別的注意使用的方式。例如，建立 2 個文字檔 doc1.txt 與 doc2.txt，其檔案內容分別只有文字 `"1"` 與 `"2"`；但檔案 doc1.txt 在輸入 `"2"` 之後並沒有按 Enter，所以只有 2 列資料；如下左所示。而檔案 doc2.txt 在輸入 `"2"` 之後按了 Enter，所以有 3 列資料；如下右所示。

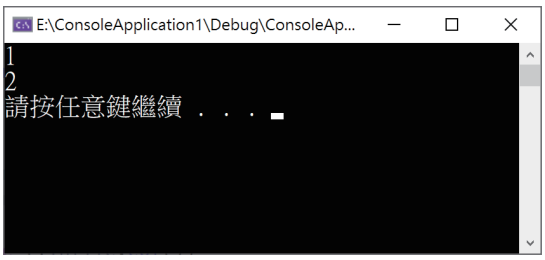




接著，使用下列程式碼讀取檔案 doc1.txt 的內容並顯示讀取的資料：

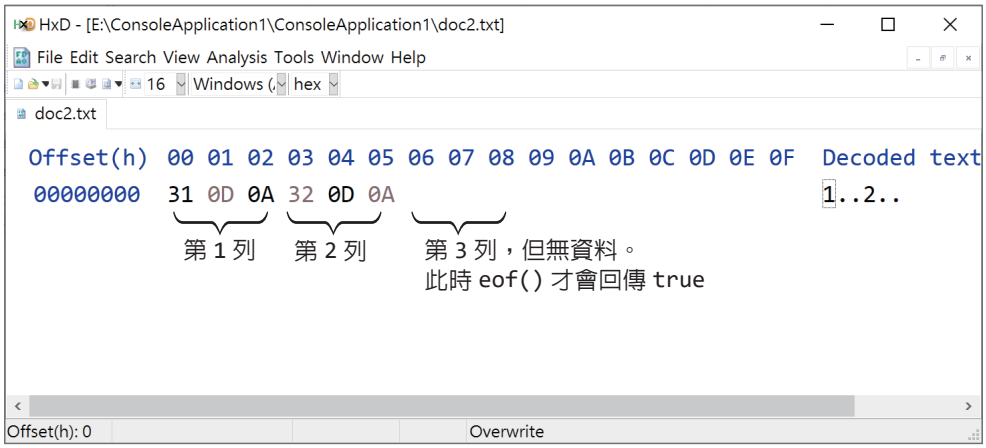
```
1 string str;
2 ifstream infile("doc1.txt", ios::in);
3
4 while (!infile.eof())
5 {
6    infile >> str;
7    cout << str << endl;
8 }
9 infile.close();
```

輸出結果如下圖左所示：正確地讀取 2 筆資料，並顯示 2 筆資料。若是將程式碼第 2 行改成開啓檔案 doc2.txt，則會多讀取 1 筆資料：最後 1 筆資料被多讀取了一次，因此顯示時也會多顯示 1 筆資料；如下圖右所示。



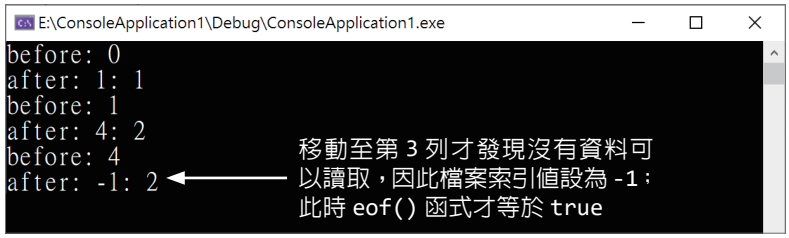
其實這並不是錯誤，而是對 "**>>**" 運算子與 **eof()** 函式運作的情形不了解所造成的誤解。因爲檔案 doc2.txt 的內容其實有 3 列，但是第 3 列並沒有資料（只是鍵入 "**2**" 後按 Enter 造成的第 3 列）。

當讀取第 2 列的資料 "**2**" 之後，因爲發現其後爲換行字元（**0D0A**），表示後面應該還有資料可以供讀取。所以回到 **while()** 敘述判斷是否已經到了檔案末端時，**eof()** 函式回傳 **false** 表示還有資料可以讀取，所以檔案索引便移動到下一個位置（第 3 列）準備讀取資料，但卻發現沒有資料可以讀取，此時 **ios::eofbit**（表示檔案已無資料）才被設定爲 **1**。但此時程式敘述第 6 行已經使用 "**>>**" 運算子要讀取資料，卻發現沒有資料；因此並沒有改變變數 **str** 原本的內容（上一次的內容等於 "**2**"），所以程式敘述第 7 行又顯示了一次 **str** 上一次的內容。



將程式的 `while()` 重複敘述區塊改爲如下，可以觀察檔案索引位置的變化。函式 `tellg()` 可以回傳目前在檔案中的索引位置。當檔案開啓時，檔案索引位置等於 `0`（第 1 個字元之前的位置）；每讀取 1 個字元則檔案索引位置加 1。

```
1  string str;
2  ifstream infile("doc2.txt", ios::in);
3
4  while (!infile.eof())
5  {
6     cout << "before: " << infile.tellg() << endl;
7     infile >> str;
8     cout << "after: " << infile.tellg() << ": " + str << endl;
9  }
10 infile.close();
```



所以，要修正此種情形，可以在使用 "`>>`" 運算子讀取資料之後，先判斷檔案的讀取是否正常，然後再繼續執行其他的程式敘述；修改後的程式如下所示。

```
1  string str;
2  ifstream infile("doc2.txt", ios::in);
3
4  while (!infile.eof())
5  {
```

```
 6      infile >> str;
 7      if (infile.good())
 8          cout << str << endl;
 9  }
10  infile.close();
```

判斷讀取資料是否成功

程式碼第 7 行使用 good() 函式判斷上一次對檔案的操作是否正常，如果正常才繼續執行第 8 行顯示讀取的資料；函式 good() 請參閱分析與討論的第 2 點。

此外，還可以直接在 while() 重複敘述中，直接使用 ">>" 運算子讀取資料，也能避免類似的情形發；如下所示：

```
1  string str;
2  ifstream infile("doc2.txt", ios::in);
3
4  while ( infile >> str )
5  {
6      cout << str << endl;
7  }
8  infile.close();
```

直接把讀取資料當成執行條件

使用 fstream/ifstream 類別的函式讀取檔案

fstream/ifstream 類別中與讀取檔案相關的函式，如下表所示。使用這些函式需要引入 iostream、fstream 與 string 此 3 個標頭檔。

函式	說明
eof()	將 ios::eofbit 設為 1，表示已經到了檔尾，並回傳 true。
gcount()	回傳上一次對檔案的讀寫成功操作的字元數，回傳值為整數型別。
get(a)	從檔案讀取 1 個字元並儲存於 a；回傳所讀取的字元。a 為字元型別。
getline(a,b[,c])	從檔案讀取 b 個字元並儲存到 a，或遇到字元 c 為止。a 為 char * 型別，b 為整數型別，c 為字元型別。
ignore(a,b)	略過檔案內 a 個字元，或遇到字元 b 為止。a 為整數型別，b 為字元型別。
peek()	檢查下一個檔索引位置是否還有資料；若有資料則回傳此字元，否則回傳 EOF。
read(a,b)	從檔案內讀取 b 個字元並儲存於 a。a 為 char * 型別，b 為整數型別。

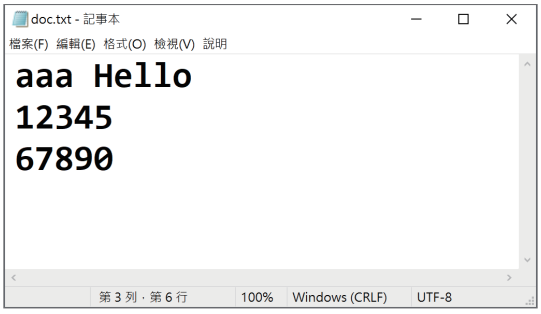
呼叫函式 eof() 時若回傳 true，表示已經到了檔案的尾端。函式 getline() 會讀取一列資料，一直遇到換行字元 '\n' 爲止；但換行字元並不會被儲存。函式 read() 會讀取指定數量的字元，並且換行字元 '\n' 也會被儲存。若在尚未讀取到指定數量的字元之前遇到了檔尾或是發生錯誤，則 ios::eofbit 與 ios::failbit 會被設定（請參閱分析與討論的第 2 點）。

練習 5：使用檔案讀取函式讀取資料

有一個檔案 doc.txt，其內容如下圖所示。寫一程式讀取此檔案，並依序完成以下操作：

1. 略過前 10 個字元，或遇到空白字元爲止。
2. 讀取 20 個字元，或遇到換行字元爲止。
3. 讀取 8 個字元，包含換行字元。
4. 檢查是否還有字元可以讀取，並逐個字元讀取檔案內剩下的所有資料。

檔案 doc.txt 內有 3 列資料。第 1 列的 "aaa" 之後有一個空白字元。第 3 列的內容 "67890" 之後並沒有按下 Enter。



解說

此練習題要從檔案 doc.txt 中讀取資料，因此使用 ifstream 類別的檔案，並且開檔模式應爲：ios::in。第 1 個操作要能指定略過數個字元，因此使用 ignore() 函式。第 2 個操作需要連續讀取指定數量的字元，並且可以指定停止讀取的字元，所以使用 getline() 函式。第 3 個操作要求能讀取換行字元，因此使用 read() 函式。

最後一個操作要讀取檔案裡剩下的資料；由於並不知道檔案中剩下多少資料，因此可以在 while() 敘述中使用 peek() 函式，先檢查檔案中是否還有資料，然後再使用 get() 函式逐一讀取資料。

執行結果

先自行在執行程式相同的路徑之下建立檔案 doc.txt 與其內容。第 1 項操作會略過 "aaa "，因此第 2 項操作便會讀取 "Hello"。第 3 項操作要讀取 8 個字元，雖然檔案的第 2 列只有 6 個字元（包含換行字元），因此也會一併讀取第 3 列的 "67" 此 2 個字元；也因爲讀取了換行字元，所以在顯示資料時會自動換行。此時，檔案內只剩下 "890" 此 3 個字元，最後一項樣操作便是使用 peek() 檢查檔案內是否還有資料可以讀取，再使用 get() 函式逐一讀取單個字元並顯示出來。

```
目前的路徑：F:\
Hello
12345
67890
```

程式碼列表

```
 1 #pragma warning(disable : 4996)
 2 #include <iostream>
 3 #include <fstream>
 4 #include <direct.h>
 5 #include <stdlib.h>
 6 using namespace::std;
 7
 8 string getExePath()
 9 {
10     string str;
11     int pos;
12
13     str = _pgmptr;
14     pos = str.find_last_of('\\');
15     if (pos == -1)
16         return str + "\\";
17     else
18         return str.substr(0, pos + 1);
19 }
20
21 int main()
22 {
23     ifstream infile;
24     char arr[20];
25     char c;
```

```
26      string str;
27
28      str = getExePath();
29      cout << " 目前的路徑：" << str << endl;
30
31      if (_chdir(str.c_str()) == -1)
32      {
33          cout << " 無法切換路徑 " << endl;
34          exit(0);
35      }
36
37      infile.open("doc.txt", ios::in);
38      if (!infile)
39      {
40          cout << " 無法開啓檔案 " << endl;
41          exit(0);
42      }
43
44      infile.ignore(10, ' ');
45
46      infile.getline(arr, 20, '\n');
47      cout << arr << endl;
48
49      infile.read(arr, 8);
50      for (int i = 0; i < infile.gcount(); i++)
51          cout << arr[i];
52
53      while (infile.peek() !=-1)
54      {
55          c = infile.get();
56          cout << c;
57      }
58
59      cout << endl;
60      infile.close();
61      system("pause");
62  }
```

程式講解

1. 程式碼第 1-6 行引入所需要的標頭檔與宣告使用 `std` 命名空間。
2. 程式碼第 8-19 行宣告自訂函式 `getExePath()`，此函式用於取得目前執行的程式的路徑。第 13 行使用全域變數 `_pgmptr` 取得執行程式的路徑，並儲存於字串變數

str。由於變數 str 中包含了路徑與檔案名稱，而此函式只想取得路徑的部分；因此，因此第 14-18 行用於判斷與尋找變數 str 中最後的 1 個目錄與檔案的分隔字元 '\'；若找得到最後 1 個分隔字元，則回傳變數 str 中包含此分隔字元與其之前的部分（路徑的部分）。

3. 程式碼第 23 行宣告 ifstream 類別的檔案變數 infile。第 24 行宣告字元型別的一維陣列 arr，陣列長度等於 20，用於儲存從檔案讀取的資料。第 25 行宣告字元變數 c，用於儲存從檔案讀取的字元。第 26 行宣告字串變數 str，用於儲存目前執行程式的路徑。
4. 程式碼第 28-29 行呼叫自訂函式 getExePath() 取的程式執行的路徑，並儲存於字串變數 str。第 31-35 行使用函式 _chdir() 切換工作路徑至目前執行程式的路徑，若切換不成功則顯示錯誤訊息並結束程式。
5. 程式碼第 37-42 行使用函式 open() 開啓檔案 doc.txt，開檔模式爲 ios::in；因此，此檔案用於讀取資料。若檔案開啓失敗則顯示錯誤訊息並結束程式。
6. 程式碼第 44 行雖然使用 ignore() 略過檔案的 10 個字元，但遇到空白字元便停止；因此，只會略過檔案資料："aaa " 此 4 個字元。
7. 程式碼第 46 行使用 getline() 函式讀取 20 個字元，但遇到換行字元即停止讀取；因此，只會讀取檔案第 1 列的 "Hello"，但並不會將換行字元儲存在陣列 arr 之中。
8. 程式碼第 49 行使用 read() 函式讀取 8 個字元，但因檔案的爲第 2 列只有 6 個字元："12345" 與 1 個換行字元，因此也會繼續讀取到第 3 列的前 2 個字元 "67"；因爲 read() 函式確實讀取到 8 個字元，因此第 50 行 for 重複敘述裡呼叫 gcount() 函式也會回傳 8。第 50-51 行使用 for 重複敘述顯示所讀取的 8 個字元。
9. 程式碼第 53-57 行用於讀取檔案內剩下的資料。第 53 行 while 重複敘述使用函式 peek() 作爲執行條件：當函式 peek() 能夠讀取檔案內的字元，則表示還能讀取資料。因此，第 55 行使用函式 get() 從檔案讀取 1 個字元。因此一共讀取 3 個字元："890"。
10. 程式碼第 60 行使用函式 close() 關閉檔案。

24-4 檔案與結構

結構可以表達複雜的資料；在儲存結構資料時，若是把結構內的變數成員逐個寫入檔案，當讀取資料時，逐一讀取資料再放入結構中，這樣的作法顯然沒有效率又麻煩。因此，能以一整個結構的形式將資料寫入檔案，或是以結構的資料形式從檔案讀取資料，這樣的檔案讀寫方式顯然更方便又有效率。

將結構寫入檔案

有一結構 _MYSTRUCT 包含 2 個成員變數：整數變數 a 與字元 b，並宣告 2 個此結構的變數 wStruct 與 rStruct；如下所示：

```
struct _MYSTRUCT
{
    int a;
    char b;
}wStruct,rStruct;
```

則將結構 wStruct 寫入檔案的方式，如下所示：

```
1 fstream file;
2
3 wStruct.a = 100;
4 wStruct.b = 'd';
5
6 file.open("doc.txt", ios::out);
7 file.write((char *)&wStruct, sizeof(wStruct));
8 file.close();
```

程式碼第 1 行宣告 fstream 類別的檔案變數 file，第 3-4 行設定結構 wStruct 的值，第 6 行以開檔模式 ios::out 建立檔案 doc.txt。第 7 行以函式 write() 將結構 wStruct 寫入檔案。函式 write() 的第 1 個參數爲結構 wStruct 的位址，第 2 個參數使用 sizeof() 函式取得結構 wStruct 的大小。

從檔案讀取結構

從檔案讀取結構的方式，如下所示。

```
1 fstream file;
2
3 file.open("doc.txt", ios::in);
4 file.read((char*)&rStruct, sizeof(rStruct));
5 file.close();
```

程式碼第 1 行宣告 fstream 類別的檔案變數 file，第 3 行以開檔模式 ios::in 開啓檔案 doc.txt。第 4 行使用函式 read() 從檔案讀取資料，並儲存於結構 rStruct。函式 read() 的第 1 個參數爲結構 rStruct 的位址，第 2 個參數使用 sizeof() 函式取得結構 rStruct 的大小。

將結構陣列寫入檔案

當多筆的結構資料以陣列的形式處理時，可以使用重複敘述從檔案中逐筆讀取結構資料，或是將結構陣列裡的資料逐筆寫入檔案；當然這樣的方式比較沒有效率。此外，也能從檔案中讀取整個結構陣列，或是將整個結構陣列寫入檔案。

▶ 逐筆將結構資料寫入檔案

有一結構 _MYSTRUCT 包含 2 個成員變數：整數變數 a 與字元 b，並宣告 2 個此結構的陣列變數 wStruct[3] 與 rStruct[3]；如下所示：

```
struct _MYSTRUCT
{
    int a;
    char b;
}wStruct[3], rStruct[3];
```

則將結構陣列 wStruct 寫入檔案的方式，如下所示：

```
1 fstream file;
2
3 file.open("doc.txt", ios::out);
4     ⋮
5 for (int i = 0; i < 3; i++)
6     file.write((char *)&wStruct[i], sizeof(wStruct[i]));
7
8 file.close();
```

程式碼第 1 行宣告 fstream 類別的檔案變數 file，第 3 行以開檔模式 ios::out 建立檔案 doc.txt。第 5-6 行使用 for 重複敘述，使用函式 write() 將結構陣列的 3 筆資料逐一寫入檔案。函式 write() 的第 1 個參數爲結構陣列第 i 筆資料 wStruct[i] 的位址，第 2 個參數使用 sizeof() 函式取得結構陣列第 i 筆資料 wStruct[i] 的大小；因爲此處是逐筆將結構寫入檔案，因此每筆結構資料的大小也能使用 sizeof(_MYSTRUCT) 表示。

▶ 一次將結構陣列寫入檔案

若結構資料的筆數並不是很多，也可以使用一次將整個結構陣列寫入檔案，如下列程式碼第 5 行所示。傳入函式 write() 的第 1 個參數是整個結構陣列 wStruct 的位址，而第 2 個參數所計算的也是整個結構陣列 wStruct 的大小。

```
1 fstream file;
2
3 file.open("doc.txt", ios::out);
4     ⋮
5 file.write((char *)&wStruct, sizeof(wStruct));
6 file.close();
```

從檔案讀取結構陣列的資料

若當時寫入檔案的資料，是以結構的方式寫入資料，則讀取資料時也能以結構的形式從檔案中讀取資料。當時寫入結構陣列資料時，可能是逐筆寫入資料或是整個結構陣列一次寫入檔案，則在讀取資料時也可以選擇逐筆讀取資料，或是一次將整個資料讀入結構陣列。

▶ 從檔案中逐筆讀取結構資料

若有多筆結構資料儲存於檔案中，則可以使用 for 重複敘述逐筆讀取資料之後再儲存於結構陣列。例如：有 3 筆結構資料儲存於檔案之中，則從檔案中讀取此 3 筆資料，如下所示。

```
1 fstream file;
2
3 file.open("doc.txt", ios::in);
4 for (int i = 0; i < 3; i++)
5     file.read((char *)&rStruct[i], sizeof(rStruct[i]));
6
7 file.close();
```

程式碼第 1 行宣告 fstream 類別的檔案變數 file，第 3 行使用開檔模式 ios::in 開啓檔案 doc.txt。第 4-5 行使用 for 重複敘與函式 read()，從檔案中讀取 3 次的資料，並儲存於結構陣列的第 i 筆資料 rStruct[i] 中。函式 read() 的第 1 個參數爲結構 rStruct[i] 的位址，第 2 個參數使用 sizeof() 函式取得結構 rStruct[i] 的大小；因爲此處是逐筆讀取資料，因此每筆結構資料的大小也能使用 sizeof(_MYSTRUCT) 表示。

▶ 從檔案中一次讀取結構陣列

若寫入到檔案的結構資料並不多時，可以一次性地讀取所有的資料並儲存到結構陣列，如程式碼第 4 行所示。傳入函式 read() 的第 1 個參數是整個結構陣列 rStruct 的位址，而第 2 個參數所計算的也是整個結構陣列 rStruct 的大小。

```
1 fstream file;
2
3 file.open("e:\\doc.txt", ios::in);
4 file.read((char *)&rStruct, sizeof(rStruct));
5
6 file.close();
```

練習 6：儲存與讀取結構陣列資料

有一個記錄學生姓名、年齡與 3 科成績的結構。寫一程式，設定 3 位學生的資料後，將此 3 位學生的資料寫入檔案。接著再從檔案讀取並顯示此 3 位學生的資料。

解說

記錄學生姓名、年齡與 3 科成績的結構，應如下形式。結構名稱爲 _STUDENT，並有 3 個變數成員：name、age 與 score；分別用於表示姓名、年齡與 3 科成績。

```
1 struct _STUDENT
2 {
3     char name[20];
4     int age;
5     float score[3];
6 }wStudent[3], rStudent[3];
```

因爲有 3 位學生，所以第 6 行定義結構之後也宣告了 2 個結構陣列變數。wStudent 儲存用於寫入檔案的學生資料，rStudent 則是儲存從檔案讀取的學生資料。若宣告檔案變數的名稱爲 file，則建立寫入資料的檔案如下第 1 行程式碼所示；第 2 行則將整個結構變數 wStudent 寫入檔案。

```
1 file.open("doc.txt", ios::out);
2 file.write((char *)&wStudent, sizeof(wStudent));
```

從檔案讀取整個資料，並儲存到結構變數 rStudent 的方式如下所示。程式碼第 1 行開啓讀取資料的檔案，第 2 行則從檔案讀取資料並儲存於結構變數 rStudent。

```
1 file.open("doc.txt", ios::in);
2 file.read((char *)&rStudent, sizeof(rStudent));
```

執行結果

```
目前的路徑：F:\
檔案寫入成功，開始讀取資料 ...
姓名 = 王小明 , 年齡 = 20
分數 1=80 分數 2=70.5 分數 3=92.345
姓名 = Brown, 年齡 = 19
分數 1=90 分數 2=91.3 分數 3=95
姓名 = 李強 , 年齡 = 21
分數 1=80 分數 2=70 分數 3=100
```

程式碼列表

```
#pragma warning(disable : 4996)
#include <iostream>
#include <fstream>
#include <direct.h>
#include <stdlib.h>
#include <string.h>
using namespace::std;

struct _STUDENT
{
    char name[20];
    int age;
    float score[3];
}wStudent[3],rStudent[3];

string getExePath()
{
    string str;
    int pos;

    str = _pgmptr;
    pos = str.find_last_of('\\');
    if (pos == -1)
        return str + "\\";
    else
        return str.substr(0, pos + 1);
}

void setData(_STUDENT student[])
{
    strcpy_s(student[0].name, " 王小明 ");
    student[0].age = 20;
    student[0].score[0] = 80;
    student[0].score[1] = 70.5;
    student[0].score[2] = 92.345;

    strcpy_s(student[1].name, "Brown");
    student[1].age = 19;
    student[1].score[0] = 90;
    student[1].score[1] = 91.3;
```

```
41      student[1].score[2] = 95;
42
43      strcpy_s(student[2].name, " 李強 ");
44      student[2].age = 21;
45      student[2].score[0] = 80;
46      student[2].score[1] = 70;
47      student[2].score[2] = 100;
48  }
49
50  int main()
51  {
52      fstream file;
53      string str;
54
55      str = getExePath();
56      cout << " 目前的路徑：" << str << endl;
57      if (_chdir(str.c_str()) == -1)
58      {
59          cout << " 無法切換路徑 " << endl;
60          exit(0);
61      }
62
63      setData(wStudent); // 設定學生資料
64
65      //========== 建立檔案、寫入資料 ===============
66      file.open("doc.txt", ios::out);
67      if (!file.is_open())
68      {
69          cout << " 無法建立檔案 " << endl;
70          exit(0);
71      }
72
73      file.write((char *)&wStudent, sizeof(wStudent));
74      file.close();
75      if (file.fail())
76      {
77          cout << " 檔案寫入失敗 " << endl;
78          exit(0);
79      }
80      else
81          cout << " 檔案寫入成功，開始讀取資料 ..." << endl;
82
```

```
83      //========== 開啓檔案、讀取資料 ===============
84      file.open("doc.txt", ios::in);
85      if (!file.is_open())
86      {
87          cout << " 無法開啓檔案 " << endl;
88          exit(0);
89      }
90
91      file.read((char *)&rStudent, sizeof(rStudent));
92      file.close();
93      if (file.fail())
94      {
95          cout << " 檔案讀取失敗 " << endl;
96          exit(0);
97      }
98      else
99      {
100         for (int i = 0; i < 3; i++)
101         {
102             cout << " 姓名 = " << rStudent[i].name << ", ";
103             cout << " 年齡 = " << rStudent[i].age << endl;
104             for (int j = 0; j < 3; j++)
105                 cout << " 分數 " << j+1 <<"="<< rStudent[i].score[j] << " ";
106             cout << endl;
107         }
108     }
109     system("pause");
110 }
```

程式講解

1. 程式碼第 1-7 行引入所需要的標頭檔與宣告使用 std 命名空間。
2. 程式碼第 9-14 行定義結構 _STUDENT。結構內有有 3 個變數成員：name、age 與 score；分別用於表示姓名、年齡與 3 科成績；並直接宣告 2 個長度等於 3 的陣列變數 wStudent 與 rStudent，分別用於儲存要寫入檔案的資料，與儲存從檔案讀取的資料。
3. 程式碼第 16-27 行宣告自訂函式 getExePath()，此函式用於取得目前執行的程式的路徑。第 21 行使用全域變數 _pgmptr 取得執行程式的路徑，並儲存於字串變數 str。由於變數 str 中包含了路徑與檔案名稱，而此函式只想取得路徑的部分；因此，第 22-26 行用於判斷與尋找變數 str 中最後的 1 個目錄與檔案的分隔字元 '\'；若找得到最後 1 個分隔字元，則回傳變數 str 中包含此分隔字元與其之前的部分（路徑的部分）。

4. 程式碼第 29-48 行宣告自訂函式 setData()，並帶有一個 _STUDENT 結構型別的陣列參數 student。第 31-35 行設定第 1 筆的學生資料；因爲結構 _STUDENT 中所使用字元陣列宣告成員變數 name；因此，第 31 行需要使用函式 strcpy_s() 將姓名字串儲存到結構的字元陣列 name。第 2、3 筆學生資料的設定都是相同的方式，如第 37-41 行與 43-47 行所示。
5. 程式碼第 52 行宣 fstream 類別的檔案變數 file。第 53 行宣告字串變數 str，用於儲存目前執行程式的路徑。
6. 程式碼第 55 行呼叫自訂函式 getExePath() 取得程式執行的路徑，並儲存於字串變數 str。第 57-61 行使用函式 _chdir() 切換工作路徑至目前執行程式的路徑，若切換不成功則顯示錯誤訊息並結束程式。
7. 程式碼第 63 行呼叫自訂函式 setData() 將學生資料填入結構陣列 wStruct。
8. 程式碼第 66-71 行使用函式 open() 開啓檔案 doc.txt，開檔模式爲 ios::out；因此，此檔案用於寫入資料。若檔案建立失敗則顯示錯誤訊息並結束程式。
9. 程式碼第 73 行使用 write() 函式把整個結構陣列 wStruct 寫入檔案 doc.txt。第 74 行使用 close() 關閉檔案。第 75-81 行使用 fail() 函式判斷剛才寫入資料是否有發生錯誤，若發生錯誤則顯示錯誤訊息並結束程式。
10. 程式碼第 84-89 行使用開檔模式 ios::in 與函式 open() 開啓檔案 doc.txt。若檔案開啓失敗則顯示錯誤訊息並結束程式。
11. 程式碼第 91 行使用 read() 函式從檔案讀取整個結構陣列資料，並儲存於結構陣列 rStudent。第 92 行使用 close() 關閉檔案。第 93-108 行爲 if…selse 判斷敘述，第 93-97 行使用 fail() 函式判斷剛才讀取資料是否有發生錯誤，若發生錯誤則顯示錯誤訊息並結束程式；否則執行 99-108 行顯示所讀取的資料。
12. 程式碼第 100-107 行使用 for 重複敘述顯示所讀取的結構陣列 rStudent 的資料。

三、範例程式解說

1. 建立專案，程式碼第 1-7 行引入所需要的標頭檔與宣告使用 std 命名空間。

```
1 #pragma warning(disable : 4996)
2 #include <iostream>
3 #include <fstream>
4 #include <direct.h>
5 #include <conio.h>
6 #include <io.h>
7 using namespace::std;
```

2. 程式碼第 9-17 行分別定義常數、結構與宣告全域變數。第 9 行定義常數 MAX_NUM 等於 5，作爲學生的總人數。第 11-15 行定義學生資料的結構 _STUDENT，包含 2 個變數成員：字元型別的一維陣列 name 用於儲存姓名、整數型別的變數 age 用於儲存年齡；並且直接宣告結構陣列變數 student，長度等於 MAX_NUM。第 17 行宣告整數變數 stutNum，用於表示目前的學生資料筆數。

```
 9 #define MAX_NUM 5  // 最大資料筆數
10
11 struct _STUDENT    // 學生資料
12 {
13     char name[20];
14     int age;
15 }student[MAX_NUM];
16
17 int stutNum = 0;   // 目前資料的筆數
```

3. 程式碼第 20-36 行爲自訂函式 setExtPath()，並帶有一個參考呼叫的參數 strPath；此函式用於取得執行程式的路徑，並將此路徑設定爲工作路徑。第 25-30 行取得執行程式的路徑，並儲存於參數 strPath。第 32-35 行將執行程式的路徑 strPath 設定爲工作路徑。若設定路徑成功則回傳 true，否則回傳 false。

```
20 bool setExePath(string &strPath)
21 {
22     string str;
23     int pos;
24
25     str = _pgmptr;
26     pos = str.find_last_of('\\');
27     if (pos == -1)
28         strPath = str + "\\";
29     else
30         strPath = str.substr(0, pos + 1);
31
32     if (_chdir(strPath.c_str()) == -1)
33         return false;
34     else
35         return true;
36 }
```

4. 程式碼第 39-79 行爲自訂函式 addData()，此函式用於新增學生資料，並將新增的資料立即儲存於檔案。第 41 行宣告 fstream 類別的檔案變數 file，第 42 行宣告 _STUDENT 結構型別的變數 stut，用於儲存所輸入的資料。第 44-48 行判斷若資料數量已經達到上限，則顯示警告訊息並返回呼叫者。

   第 51-53 行顯示提示訊息與讀取輸入的姓名，並儲存於結構的變數成員 stut.name。第 54-58 行判斷若輸入的姓名等於 "-1"，則顯示訊息並返回呼叫者。第 60-61 行讀取輸入的年齡並儲存於結構的變數成員 stut.age。

```
39 void addData()
40 {
41     fstream file;
42     _STUDENT stut;
43
44     if (stutNum >= MAX_NUM)
45     {
46         cout << " 已達資料上限 " << endl;
47         return;
48     }
49
50     //--------- 輸入資料 -----------
51     cout << endl;
52     cout << " 輸入姓名 ( 輸入 -1 結束 )：";
53     cin >> stut.name;
54     if (strcmp(stut.name, "-1")==0)
55     {
56         cout << " 結束輸入，";
57         return;
58     }
59
60     cout << " 輸入年齡：";
61     cin >> stut.age;
```

   第 64-66 行使用函式 open() 開啓檔案 doc.txt，開檔模式爲 ios::out|ios::app；因此，此檔案用於附加資料。第 65-66 行判斷若檔案開啓失敗則顯示錯誤訊息，否則執行第 68-78 行將結構變數 stut 寫入檔案。第 69-76 行將結構陣列 stut 一次寫入檔案，並使用函式 fail() 判斷資料是否成功寫入檔案；若資料成功寫入檔案，則第 75 行將學生資料的筆數 stutNum 加 1。第 77 行使用 close() 函式關閉檔案。

```
63      //--------- 寫入資料 ------------
64      file.open("doc.txt", ios::out | ios::app);
65      if (!file.is_open())
66          cout << " 無法開啓檔案 " << endl;
67      else
68      {
69          file.write((char*)& stut, sizeof(stut));
70          if (file.fail())
71              cout << " 資料寫入失敗 " << endl;
72          else
73          {
74              cout << " 資料寫入成功 " << endl;
75              stutNum++;  // 資料筆數增加 1 筆
76          }
77          file.close();
78      }
79  }
```

5. 程式碼第 82-105 行爲自訂函式 `readData()`，此函式用於從檔案中讀取資料，並儲存於結構。第 84 行宣告 `fstream` 類別的檔案變數 `file`，第 86 行先使用函式 `_access()` 判斷檔案 `doc.txt` 是否存在，若檔案存在則執行第 87-104 行的程式敘述。第 88 行使用 `open()` 函式與開檔模式 `ios::in` 開啓檔案 `doc.txt`，第 89 行使用 `is_open()` 函式判斷若檔案 `doc.txt` 可以順利開啓，則執行第 92-103 行。

   第 93 行在讀取檔案資料之前，先將記錄資料筆數的變數 `stutNum` 設定爲 `0`，表示尚未從檔案讀取任何資料。第 94-101 行使用 `while()` 重複敘述讀取檔案內的資料，執行條件爲 `!file.eof()`；因此，`while()` 重複敘述會持續從檔案中讀取資料，直到檔案內已無資料可供讀取爲止。第 96 行使用 `read()` 函式從檔案中讀取資料，並儲存於 `student` 結構陣列的第 `stutNum` 的位置。第 97-100 行使用 `fail()` 函式判斷讀取資料是否成功，若讀取失敗則離開 `while` 重複敘述，若讀取資料成功則將資料筆數 `stutNum` 加 `1`。當所有資料都讀取完畢，則執行第 102 行使用 `close()` 關閉檔案。

```
82  void readData()
83  {
84      fstream file;
85
86      if (_access("doc.txt",0)!=-1) // 判斷檔案是否存在
87      {
88          file.open("doc.txt", ios::in);  // 開啓檔案
89          if (!file.is_open())
```

```
90              cout << " 無法開啓檔案 " << endl;
91          else
92          {
93              stutNum = 0;  // 資料筆數先設定爲 0
94              while(!file.eof())
95              {
96                  file.read((char*)& student[stutNum], sizeof(_STUDENT));
97                  if (file.fail())
98                      break;
99
100                 stutNum++; // 資料筆數累加 1 筆
101             }
102             file.close();
103         }
104     }
105 }
```

6. 程式碼第 108 行爲自訂函式 showData()，此函式用於顯示學生的資料。第 110-111 行判斷若記錄學生資料筆數的變數 stutNum 等於 0，表示目前沒有學生資料；否則執行第 113-118 行使用 for 重敘述顯示 stutNum 筆的學生資料。

```
108 void showData()
109 {
110     if (stutNum == 0)
111         cout << " 無資料 " << endl;
112     else
113         for (int i = 0; i < stutNum; i++)
114         {
115             cout << endl;
116             cout << " 姓名 = " << student[i].name << endl;
117             cout << " 年齡 = " << student[i].age << endl;
118         }
119 }
```

7. 程式碼第 122-129 行爲自訂函式 showMenu()，此函式用於顯示功能選單。

```
122 void showMenu()
123 {
124     system("cls");
125     cout << "1. 新增資料 " << endl;
126     cout << "2. 顯示資料 " << endl;
127     cout << "3. 結束 " << endl;
128     cout << " 輸入選擇 (1-3): ";
129 }
```

8. 開始於 main() 主函式中撰寫程式。程式碼第 134 行宣告字串變數 strPath，用於儲存執行程式的路徑。第 135 行宣告整數變數 sel，用於儲存使用者所輸入的選項代號。第 137-141 行呼叫自訂函式 setExtPath()，並傳入引數 strPath；若取得執行程式的路徑之後，會將路徑儲存於變數 strPath。若無法成功切換路徑，則顯示錯誤訊息並結束程式。第 143 行呼叫自訂函式 readData() 讀取資料；如此才能知道已經儲存了多少筆的學生資料資料，變數 stutNum 才會被設定正確的值。

```
134 string strPath;
135 int sel;
136
137 if (!setExePath(strPath))
138 {
139     cout << " 無法切換工作路徑 " << endl;
140     exit(0);
141 }
142
143 readData(); // 讀取資料
```

第 145-168 行爲 while 無窮迴圈，第 147 行呼叫自訂函式 showMenu() 顯示選單，第 148 行取得使用者所輸入的選項，並儲存於變數 sel。第 150-165 行爲 switch…case 選擇敘述，根據變數 sel 的值執行相對應的功能。

第 152-153 行爲選擇了「新增資料」，呼叫自訂函式 addData() 執行此功能。第 155-158 行爲選擇了「顯示資料」，呼叫自訂函式 readData() 與 showData()，分別讀取資料與顯示資料。程式碼第 160-161 行爲「結束」的功能，呼叫函式 exit() 結束程式。第 163-164 行爲 default 區塊；若輸入錯誤的選項則第 163 行顯示錯誤訊息。第 166-167 行等待使用者按任一鍵後繼續執行。

```
145 while (true)
146 {
147     showMenu();
148     cin >> sel;
149
150     switch (sel)
151     {
152         case 1: addData();
153             break;
154
155         case 2:
```

```
156                 readData();
157                 showData();
158                 break;
159
160           case 3: exit(0);
161                 break;
162
163           default: cout << " 輸入錯誤，";
164                 break;
165       }
166       cout << " 按鍵繼續 ...";
167       while (!_kbhit());
168  }
169
170  system("pause");
```

重點整理

1. fstream、ofstream 與 ifstream 此 3 個類別都能處理檔案的讀寫；但需要配合正確的開檔模式才能有作用。
2. 可以使用 ">>" 與 "<<" 運算子，分別從檔案讀取資料與把資料寫入檔案。
3. 從檔案讀取資料與把資料寫入檔案，有各自專用的函式；這些函式有不同的使用時機。
4. 結構資料能夠以結構的形式直接寫入檔案；也能夠以結構的形式從檔案直接讀取資料。

分析與討論

1. 開檔模式需和讀寫檔案的函式搭配，否則並不會有正確的輸出入結果。例如：

```
1  fstream file;
2
3  file.open("doc.txt", ios::in );
4  file << " 使用 fstream 寫入資料 ";
5  file.close();
6
```

   程式碼第 3 行使用的開檔模式爲 ios::in，因此檔案用於讀取資料。但是第 4 行卻使用 "<<" 運算子將資料寫入檔案；因此造成了矛盾。此種情形 C++ 編譯程式並不會發出錯誤，程式也能正常執行，只是無法如預期地能從檔案讀取資料或是寫入資料。

2. 檔案處理過程中，可以由經由呼叫函式 good()、eof()、fail() 與 bad() 得知檔案的操作是否正常或是發生錯誤。在 basic_ios 類別裡，定義了 iostate 型別的常數，用於表示串流的狀態：goodbit、eofbit、failbit 與 badbit；如下表所列。表中也列出了不同的 iostate 常數對於不同函式的回傳值情形。

iostate 的常數	說明	bad()	eof()	fail()	good()	rdstate()
ios::goodbit	操作正常				true	ios::goodbit
ios::eofbit	到了檔案尾端		true			ios::eofbit
ios::failbit	邏輯操作錯誤			true		ios::failbit
ios::badbit	讀寫操作錯誤	true		true		ios::badbit

此 4 個常數可以使用函式 rdstate() 讀取；例如：判斷串流 file 上一次操作是否成功，如下所示：

```
if(file.rdstate() & ios::goodbit)
    ⋮
```

或者也可以使用函式 good()，如下所示：

```
if (file.good() == true)
    ⋮
```

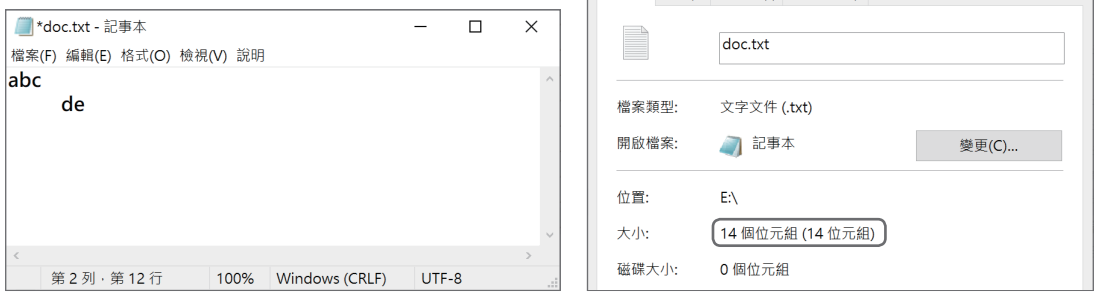
因此，像是在分析與討論的第 1 點所說的問題，雖然 C++ 的編譯過程沒有問題、也能執行；但並有任何結果。此時，便可以使用函式 fail() 捕捉到錯誤；如下所示。

```
1 fstream file;
2
3 file.open("doc.txt", ios::in );
4 file << " 使用 fstream 寫入資料 ";
5 if (file.fail())
6     ⋮
7 file.close();
```

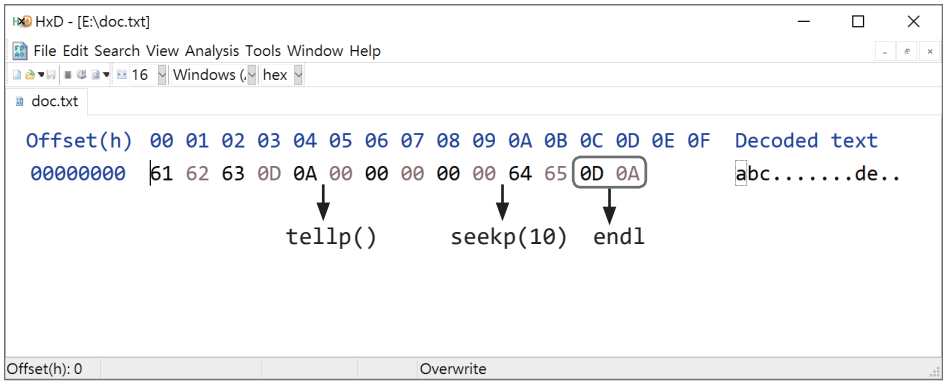
3. ofstream 類別有 2 個函式 tellp() 與 seekp()，分別用於回傳目前在檔案中的索引位置，以及設定在檔案中的索引位置。例如以下範例：程式碼第 1 行宣告 ofstream 類別的檔案變數 file，並使用開檔模式 ios::out 建立檔案 doc.txt。第 3 行將字串 "abc" 與 endl 寫入檔案。第 4 行使用 tellp() 顯示目前在檔案中的位置：5。第 5 行先使用 seekp() 將寫入位置移至 10，然後第 6 行再將字串 "de" 與 endl 寫入檔案。

```
1 ofstream file("doc.txt", ios::out);
2
3 file << "abc" << endl;
4 cout << file.tellp() << endl;
5 file.seekp(10);
6 file << "de" << endl;
7
8 file.close();
```

若用記事本開啓 doc.txt，則內容如下圖左所示：



使用可以將文件內容以 16 進位呈現的軟體觀察檔案 doc.txt；如下圖所示，可以發現 `endl` 寫入檔案後，實際上是儲存 `0D0A` 此 2 個位元組。因此，第 3 行程式敘述實際上是寫入 5 個位元組到檔案；所以，`tellp()` 的回傳值等於 5。而 `seekp(10)` 則是從檔案的開頭往後移動到 10 個位元組的位置，並且中間沒有資料的地方以 `00` 填滿。因此，此檔案的長度等於 14 個位元組，如上圖右所示。



函式 seekp() 可以指定從檔案的何處開始移動，如下表所示。

常數	說明
ios_base::beg	從檔案開頭
ios_base::cur	從目前的位置
ios_base::end	從檔案的尾端

例如，設定從檔案開頭移動 10 個位元組：

```
file.seekp(10, ios_base::beg);
```

或是：

```
file.seekp(10, ios::beg);
```

4. ifstream 類別也有 2 個函式 tellg() 與 seekg()，分別用於回傳目前在檔案中的位置，以及設定在檔案中的位置；其使用方式與 tellp() 和 seekp() 相同。
5. 結構內的字串變數，建議使用字元陣列來宣告字串；如此每個寫入檔案內的結構大小才會一樣，從檔案中讀取結構資料時才不至於發生錯誤。

   若在結構中以 string 型別宣告字串變數，則所儲存的字串資料長度不一樣時，也造成每筆結構資料的大小不同；如此一來，把這些結構資料寫入檔案之後，再從檔案中讀取資料時容易發生錯誤。

程式碼列表

```
 1 #pragma warning(disable : 4996)
 2 #include <iostream>
 3 #include <fstream>
 4 #include <direct.h>
 5 #include <conio.h>
 6 #include <io.h>
 7 using namespace::std;
 8
 9 #define MAX_NUM 5  // 最大資料筆數
10
11 struct _STUDENT    // 學生資料
12 {
13     char name[20];
14     int age;
15 }student[MAX_NUM];
16
```

```
17 int stutNum = 0;   // 目前資料的筆數
18
19 //---- 設定執行程式的路徑爲工作路徑 -------
20 bool setExePath(string &strPath)
21 {
22     string str;
23     int pos;
24
25     str = _pgmptr;
26     pos = str.find_last_of('\\');
27     if (pos == -1)
28         strPath = str + "\\";
29     else
30         strPath = str.substr(0, pos + 1);
31
32     if (_chdir(strPath.c_str()) == -1)
33         return false;
34     else
35         return true;
36 }
37
38 //------- 新增資料 --------------
39 void addData()
40 {
41     fstream file;
42     _STUDENT stut;
43
44     if (stutNum >= MAX_NUM)
45     {
46         cout << " 已達資料上限 " << endl;
47         return;
58     }
59
50     //--------- 輸入資料 -----------
51     cout << endl;
52     cout << " 輸入姓名 ( 輸入 -1 結束 )：";
53     cin >> stut.name;
54     if (strcmp(stut.name, "-1")==0)
55     {
56         cout << " 結束輸入，";
57         return;
58     }
```

```
59
60      cout << " 輸入年齡：";
61      cin >> stut.age;
62
63      //--------- 寫入資料 ------------
64      file.open("doc.txt", ios::out | ios::app);
65      if (!file.is_open())
66          cout << " 無法開啓檔案 " << endl;
67      else
68      {
69          file.write((char*)& stut, sizeof(stut));
70          if (file.fail())
71              cout << " 資料寫入失敗 " << endl;
72          else
73          {
74              cout << " 資料寫入成功 " << endl;
75              stutNum++;  // 資料筆數增加 1 筆
76          }
77          file.close();
78      }
79  }
80
81  //-------- 讀取資料 -----------------
82  void readData()
83  {
84      fstream file;
85
86      if (_access("doc.txt",0)!=-1) // 判斷檔案是否存在
87      {
88          file.open("doc.txt", ios::in);  // 開啓檔案
89          if (!file.is_open())
90              cout << " 無法開啓檔案 " << endl;
91          else
92          {
93              stutNum = 0;  // 資料筆數先設定爲 0
94              while(!file.eof())
95              {
96                  file.read((char*)& student[stutNum], sizeof(_STUDENT));
97                  if (file.fail())
98                      break;
99
100                 stutNum++; // 資料筆數累加 1 筆
```

```
101             }
102             file.close();
103         }
104     }
105 }
106
107 //---------------------------------------
108 void showData()
109 {
110     if (stutNum == 0)
111         cout << "無資料" << endl;
112     else
113         for (int i = 0; i < stutNum; i++)
114         {
115             cout << endl;
116             cout << "姓名 = " << student[i].name << endl;
117             cout << "年齡 = " << student[i].age << endl;
118         }
119 }
120
121 //-------- 顯示選單 -----------------
122 void showMenu()
123 {
124     system("cls");
125     cout << "1. 新增資料" << endl;
126     cout << "2. 顯示資料" << endl;
127     cout << "3. 結束" << endl;
128     cout << "輸入選擇 (1-3): ";
129 }
130
131 //-----------------------------------
132 int main()
133 {
134     string strPath;
135     int sel;
136
137     if (!setExePath(strPath))
138     {
139         cout << "無法切換工作路徑" << endl;
140         exit(0);
141     }
142
```

```
143     readData(); // 讀取資料
144
145     while (true)
146     {
147         showMenu();
148         cin >> sel;
149
150         switch (sel)
151         {
152             case 1: addData();
153                 break;
154
155             case 2:
156                 readData();
157                 showData();
158                 break;
159
160             case 3: exit(0);
161                 break;
162
163             default: cout << " 輸入錯誤，";
164                 break;
165         }
166         cout << " 按鍵繼續 ...";
167         while (!_kbhit());
168     }
169
170     system("pause");
171 }
```

本章習題

1. 在當前執行程式的路徑之下，建立檔案 doc.txt，並顯示建立成功或失敗的訊息。先判斷此檔案是否存在，若此檔案已經存則先刪除檔案後再建立新檔。
2. 接續第 1 題，可以將使用者輸入的資料儲存至檔案；當輸入 -1 則結束程式。
3. 接續第 2 題，開啓此檔案並顯示檔案內的所有內容。
4. 修改第 1 題，若檔案已經存在，則詢問是否要刪除檔案並建立新檔案，或是開啓舊檔繼續新增資料。
5. 接續上述 4 題，替這些功能新增功能選單。
6. 修改本範例，新增刪除指定學生的資料。
7. 修改本範例，新增修改指定學生的資料。