C++程式數計：檔案處裡

目錄

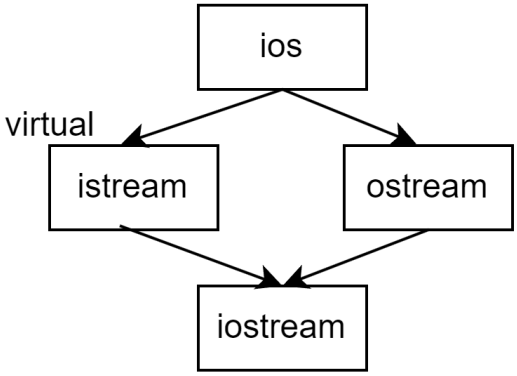
1. 資料流

資料流代表資料從起源（Source）流向終點（Sink）的一種傳輸通道。

在 C++ 中，所有輸入與輸出皆是以資料流方式運作。資料可以從檔案、鍵盤輸入，流向記憶體、顯示器等裝置。

C++ 提供多種資料流類別，用來統一處理不同裝置的輸入與輸出，避免開發者手動處理各種設備格式差異:

- ios 類別：所有資料流的基礎類別，定義了基本的 I/O 功能與格式控制等。
- istream 類別：支援輸入資料流，定義輸入資料的格式與動作。
  - ifstream 類別：支援從檔案讀取資料（file → memory）
- ostream 類別：支援輸出資料流，定義輸出資料的格式與動作。
  - ofstream 類別：支援將資料寫入檔案（memory → file）
- iostream 類別：同時支援輸入與輸出（繼承 `istream` 和 `ostream`）
  - fstream 類別：同時支援檔案的讀取與寫入



2. 檔案（File）

檔案是儲存資料的單位，資料會儲存在非揮發性（Non-volatile）媒體中，例如：HDD/SSD/隨身碟/硬碟。

檔案中包含：資料本體、日期、權限、隱藏等屬性。每個檔案都有一個檔案名稱（File Name）：

副檔名	檔案類型
.h, .hpp, .hxx	C++ 標頭檔
.cpp, .cxx, .cc	C++ 原始程式檔
.gif	圖檔（GIF 格式）
.zip	壓縮檔（ZIP 格式）
.doc	Microsoft Word 文件
.html, .htm	網頁檔

3. 檔案分類

3.1 文字檔（Text File）：

- 以字元編碼方式儲存。
- 例如 Windows 的記事本（Notepad）會預設使用 ASCII 編碼。
- 每個字元占用一組位元組（Byte）。範例：存入 1234567890 → 10 個字元 → 需要 10 位元組空間。

3.2 二進位檔（Binary File）：

- 以二進位格式直接儲存記憶體中的原始資料。
- 優點：儲存快速、省空間、隨機存取效率高。適合儲存程式檔、圖片、影音等

3.3 檔案存取方式

循序式存取（Sequential Access）：從頭到尾「一筆一筆」依序存取資料。

- 資料會被附加（append）到檔案尾端。
- 常用於文字檔，亦稱為循序檔。

隨機式存取（Random Access）

- 可直接指定檔案中的任意位置讀寫資料。
- 適合需要頻繁修改資料的應用（例如資料庫、帳戶紀錄）。通常使用二進位檔來實作此方式，效率高。

4. 檔案的輸出入管理

C++ 檔案輸出入比 C 更簡單，透過 fstream 標頭檔可簡單管理檔案資料流

使用上述提到的 ifstream、ofstream、fstream。

4.1 檔案開啟

```
#include <fstream>
#include <iostream>
```

```cpp
using namespace std;

int main() {
    ifstream fin;
    fin.open("testFile.txt", ios::in); // 開啟檔案

    if (!fin.is_open()) {
        cout << "檔案無法開啟!!" << endl;
    } else {
        cout << "檔案開啟..." << endl;
        cout << "開啟資料流..." << endl;
        fin.close(); // 關閉檔案
    }
    return 0;
}
```

ios 開啟模式：

模式	說明
ios::in	以讀取模式開檔，檔案若不存在會錯誤
ios::out	以寫入模式開檔，會覆蓋舊資料
ios::app	以附加模式開檔，資料會寫到檔案結尾
ios::ate	開檔後游標定位在檔尾，可移動讀寫位置
ios::trunc	若檔案存在，先清空內容再寫入
ios::binary	以二進位模式操作（非純文字檔）

4.2 文字檔寫入

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

int main() {
    ofstream fileOutput;
    fileOutput.open("fileOutput.txt", ios::out);

    if (!fileOutput.is_open()) {
        cout << "檔案開啟錯誤！" << endl;
        return 1;
    } else {
        fileOutput << "今日事今日畢" << endl;
        fileOutput << "2025.07.05 沒有世界末日" << endl;
```

```
    }
    fileOutput.close();
    return 0;
}
```

4.3 文字檔讀取

方法一：抽取運算子（>>）：從檔案中依據資料型別，讀取一個字串或數值到變數中

方法二：檔案物件.get(變數)：從檔案中一次讀取一個字元，包含空白與換行

方法三：檔案物件.getline(字串陣列, 陣列大小)：從檔案中讀取一整行文字，直到遇到 \n 為止，並存入 str 陣列中。

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

int main() {
    // 宣告變數
    string str;
    char data[100];
    char oneChar;

    // 建立輸入檔案串流物件
    ifstream fin;
    fin.open("text1.txt");  // 開啟檔案

    //① 使用 get() 讀取前 12 個字元
    for (int i = 0; i < 12; i++) {
        fin.get(oneChar);
        cout << oneChar;
    }

    //② 使用 getline() 讀取接下來的一整行
    fin.getline(data, sizeof(data));
    cout << data << endl;

    //③ 使用 >> 運算子讀取剩餘字串（到檔尾）
    while (!fin.eof()) {
        fin >> str;
        cout << str << endl;
    }
```

```
    // 關閉檔案
    fin.close();
    return 0;
}
```

4.4 二進位檔的讀取

```
檔案物件.read((char*) &變數, sizeof(變數));

// 將變數的位址轉為 char* 型態傳入
// 透過 sizeof(變數) 指定要讀取的位元組數
// 適合結構化或定長格式的資料讀取（如學號、電話、座標等）
```

```
#include <iostream>   // 引入標準輸出
#include <fstream>    // 引入檔案處理
using namespace std;

int main() {
    ifstream fileInput;       // 建立讀取檔案物件
    char str[8];              // 姓名最多 7 字元 + '\0'
    int num;                  // 電話號碼為整數

    fileInput.open("text2.txt", ios::binary | ios::in); // 以二進位模式開檔

    if (!fileInput.is_open()) {
        cout << "檔案開啟錯誤！" << endl;
        return 1;
    } else {
        cout << "姓名\t電話" << endl;
        cout << "=================" << endl;

        // 讀取第一組資料
        fileInput.read(str, sizeof(str));
        fileInput.read((char*)&num, sizeof(int));

        // 進入迴圈讀取後續資料
        while (!fileInput.eof()) {
            cout << str << "\t" << num << endl;

            fileInput.read(str, sizeof(str));
```

```
            fileInput.read((char*)&num, sizeof(int));
        }
    }

    fileInput.close();  // 關閉檔案
    return 0;
}
```

5. 隨機存取模式

C++ 提供的隨機存取函式（針對 ifstream / ofstream）

函數	說明
seekg(pos)	將讀取指標移動到檔案內的某個位置（get）
seekg(pos, dir)	指定起始位置為 beg、cur、end，再偏移 pos
tellg()	取得目前讀取指標的位置（get）
seekp()	寫入用（put）版本
tellp()	取得目前寫入指標的位置

方向（seek_dir）參數選擇

- ios::beg：檔案開頭位置
- ios::cur：目前指標位置
- ios::end：檔案結尾位置

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

class NOTE {
protected:
    char str[8];   // 姓名
    int num;       // 電話
public:
    void showNote() {
        cout << "姓名：" << str << endl;
        cout << "電話：" << num << endl;
    }
};

int main() {
    NOTE myNOTE;
```

```
    int noteLength = sizeof(myNOTE);  // 每筆資料大小（固定長度）
    int n;

    ifstream fileInput("text2.txt", ios::binary | ios::in);  // 二進位開檔
    if (!fileInput.is_open()) {
        cout << "檔案開啟錯誤！" << endl;
        return 1;
    }

    cout << "請問要讀取第幾筆資料？";
    cin >> n;

    fileInput.seekg((n - 1) * noteLength, ios::beg);  // 計算第 n 筆位置並跳轉
    fileInput.read((char*)&myNOTE, noteLength);       // 讀取資料

    cout << "第 " << n << " 筆資料如下：" << endl;
    myNOTE.showNote();

    fileInput.close();
    return 0;
}
```