

# [C++] Loading Files

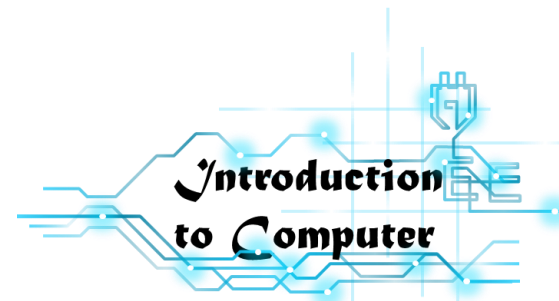
TA: 電子一 謝明倫

yans@media.ee.ntu.edu.tw

2014/02/27

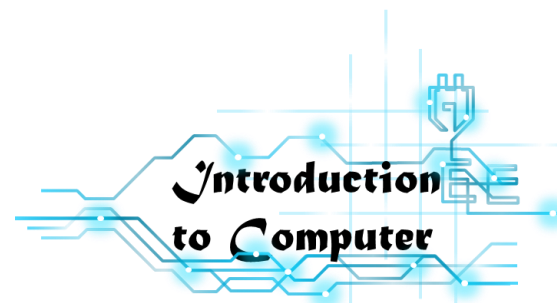


臺灣大學



# 目錄

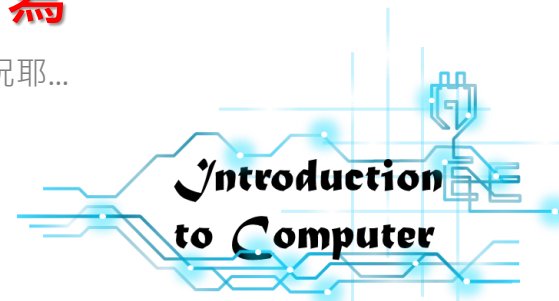
- 讀寫檔案模式
- C++ library 讀/寫檔方法
- 附註



# 讀寫檔案模式

- 除了讀檔(in)與寫檔(out)的分別之外，還另有一些模式。其中最重要的，是區分
  - Text(文字模式)      ← default(預設選項)
  - Binary(原始資料)
- 差異:Text模式會視當前的處理系統對內容的解釋做**修改**，Binary模式則是純粹拿到原始資料。
  - 差異尤其發生在“換行”上, (可自行google: windows linux new line)
  - 所以好好一張圖選錯模式開**可能讀不到or多讀到**某些資料
- **基本原則: 非文字檔一率使用Binary模式讀/寫**

所以文字檔用text囉? 不一定...看狀況耶...



# C++ library 讀/寫檔

- 利用fstream (ifstream讀, ofstream寫)
  - .open(const char\* filename, ios\_base::openmode mode)

constant	access
ios::in	File open for reading.
ios::out	File open for writing.
ios::binary	Operations are performed in binary mode rather than text.

```
#include<fstream>
```

```
...
```

```
ifstream pic1;
```

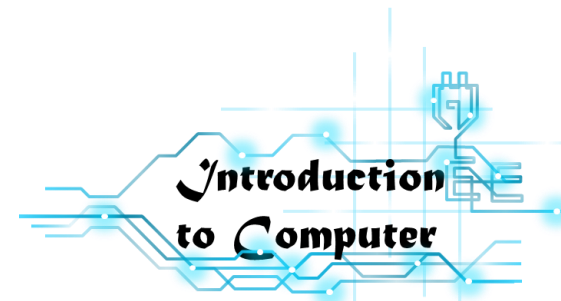
```
ofstream pic2;
```

```
pic1.open(picPath, ios::in); // 讀檔且text模式
```

```
pic1.open(picPath, ios::in | ios::binary); // 讀檔且binary模式
```

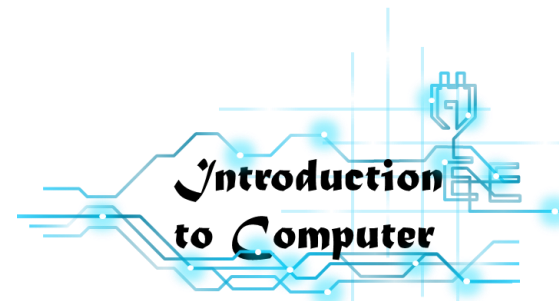
```
pic2.open(picPath, ios::out); // 寫檔且text模式
```

```
pic2.open(picPath, ios::out | ios::binary); // 寫檔且binary模式
```

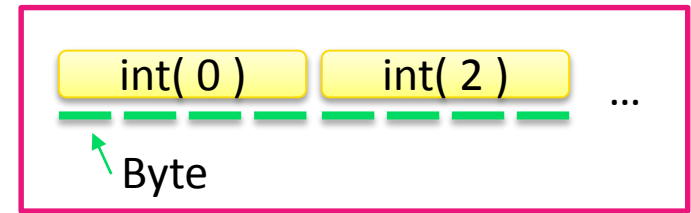


# C++ library 讀/寫檔

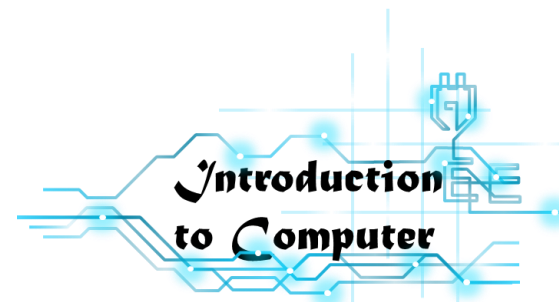
- 開啟檔案後...相關的輸入輸出函式都與iostream相同
  - 讀(in)
    - **operator<>>**: 用法跟大家用cin時一模一樣
    - 進階函式(read, getline...)
  - 寫(out)
    - **operator<<**: 用法跟大家用cout時一模一樣
    - 進階函式(write, put...)
- 執行完記得.close()以關閉檔案



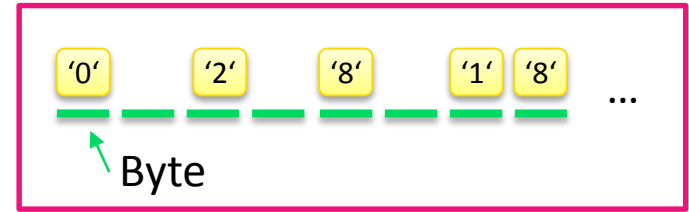
# Example (Binary)



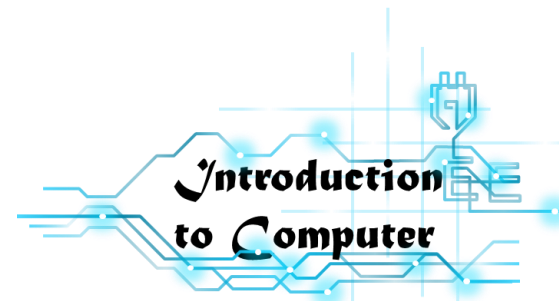
```
1  #include<iostream>
2  #include<fstream>
3  using namespace std;
4  int main(){
5      ofstream fout;
6      fout.open("file.xx",ios::out|ios::binary);
7      for(int i=0;i<10;++i)
8          fout << 2*i*i;
9      fout.close();
10
11     ifstream fin;
12     fin.open("file.xx",ios::in|ios::binary);
13     for(int i=0,temp;i<10;++i){
14         fin >> temp;
15         cout << temp << endl;
16     }
17     fin.close();
18     return 0;
}
```



# Example (Text)

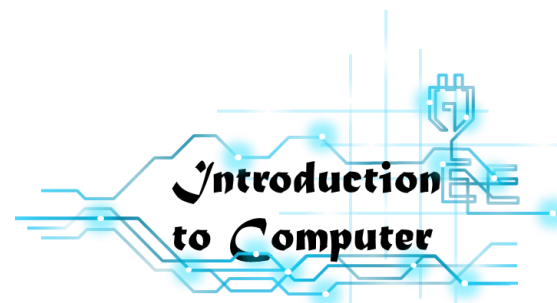


```
1  #include<iostream>
2  #include<fstream>
3  using namespace std;
4  int main(){
5      ofstream fout;
6      fout.open("file.xx",ios::out);
7      for(int i=0;i<10;++i)
8          fout << 2*i*i << ' ';
9      fout.close();
10
11     ifstream fin;
12     fin.open("file.xx",ios::in);
13     for(int i=0,temp;i<10;++i){
14         fin >> temp;
15         cout << temp << endl;
16     }
17     fin.close();
18     return 0;
}
```



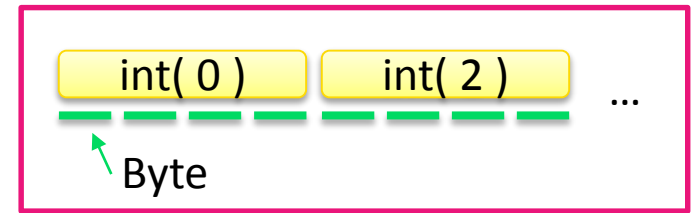
# C++ library 讀/寫檔 - 稍微進階一點點

- 一樣準備好ifstream/ofstream之後
  - 讀(in)
    - **.read(某個pointer, size)**: 自檔案讀取size這麼多bytes到pointer裡面
    - 詳細說明請自行參照文件istream
      - <http://www.cplusplus.com/reference/istream/istream/read/>
  - 寫(out)
    - **.write(某個pointer, size)**: 自pointer讀取size這麼多bytes寫到檔案裡面
    - 詳細說明請自行參照文件ostream
      - <http://www.cplusplus.com/reference/ostream/ostream/write/>

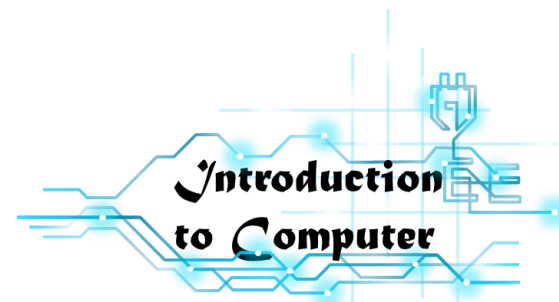




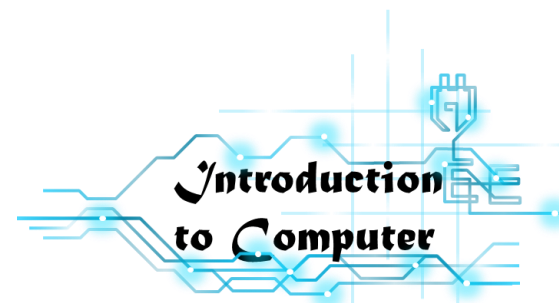
# Example (Binary)



```
1  #include<iostream>
2  #include<fstream>
3  using namespace std;
4  int main(){
5      int arr[10],arr2[10];
6      for(int i=0;i<10;++i) arr[i]= 2*i*i;
7      ofstream fout;
8      fout.open("file.xx",ios::out|ios::binary);
9      fout.write( (char*)arr, 10*sizeof(int) );
10     fout.close();
11
12     ifstream fin;
13     fin.open("file.xx",ios::in|ios::binary);
14     fin.read( (char*)arr2, 10*sizeof(int) );
15     fin.close();
16     for(int i=0,temp;i<10;++i)
17         cout << arr2[i] << endl;
18     return 0;
}
```



附註



# 附註

- 看到4Bytes數字型資料請多利用`fin>> unsigned int`
  - 如果讀成4個char或unsigned char的話可能會有Little Endian問題
  - [http://en.wikipedia.org/wiki/Little\\_Endian](http://en.wikipedia.org/wiki/Little_Endian)
- 看到2Bytes數字型資料請多利用`fin>> unsigned short`
  - 除了第一筆是兩個char(非數字型)
- 後面pixel資料用unsigned char array來接

Shift	Name	Size
0x00	Identifier (ID)	2
0x02	File Size	4
0x06	Reserved	4
0x0A	Bitmap Data Offset	4
0x0E	Bitmap Header Size	4
0x12	Width	4
0x16	Height	4
0x1A	Planes	2
0x1C	Bits Per Pixel	2



# 附註

- C library請自行參考↓本文不說明之
  - <http://www.cplusplus.com/reference/cstdio/?kw=cstdio>
  - binary讀寫檔案請在fopen的mode裡多加“b”

