1、字节流与有格式的I/O

C++以字节流的形式实现基于控制台和基于文件的输入输出（即 I/O），即数据按照内存地址的顺序逐个字节在内存和输入输出设备之间传递。
有格式的 I/O 操作以对数据类型敏感的方式执行。例如对于 cout<<(x=65);这一语句，如果 x 是 int 类型，则会显示 65；如果 x 是 char 类型，则会显示 A。由于需要额外的时间来根据数据类型处理字节内容，因此有格式的 I/O 操作不适于处理大容量的数据传输。

2、输入输出缓冲区

对于基于控制台的输入操作，用户按下回车键后会将其输入的字节保存在 cin 的缓冲区里，在执行 cin>>、cin.get 等读取 cin 缓冲区的操作时才将这些字节所表示的数据存入变量。

对于输出操作，只有在刷新输出流对象的缓冲区时才会将其中的字节传递到输出设备上。
刷新输出流缓冲区的时机：
① 程序正常结束（即 main 函数返回）时；
② 当缓冲区已满且要写入下一个值之前；
③ 执行到标准库的输出流操纵符（如行结束符 endl）时；
④ 当输出流与输入流关联且读取输入流时（在标准库中，cout 和 cin 被关联在一起，因此每个使用 cin 的输入操作都将刷新 cout 的缓冲区）。

3、get函数的三个重载版本和getline函数

输入流成员函数 get 有三个重载版本，分别是无参数的、一个参数的和三个参数的。
① 无参数的 get 返回读取的一个字符。
② 一个参数的 get 返回当前对象，并把从缓冲区读取的字符存入参数（参数为引用传递）。类似的，输出流成员函数 put 也返回当前对象，并把参数放入缓冲区。
③ 三个参数的 get 和 getline 类似，但前者将结束符保留在缓冲区里，后者则删掉结束符。

4、文件结束符EOF和输入流成员函数eof

EOF 是表示文件结束的字符，但在控制台上也可以输入（在 Windows 系统中是 Ctrl+z）。
可以用读取单个字符的方式读到 EOF，因此可以根据读到的字符是否为 EOF 来判断文件是否结束；也可以用>>读到 EOF，此时>>操作会返回 0。
不管用哪种方式读到 EOF，都会使输入流成员函数 eof 的返回值变成 1（否则返回 0）。

5、文件的概念、打开与关闭

C++把文件也当作字节流，只不过它位于硬盘（或其它外存）中。一个文件无论有多少行，都可以当作一行用换行符分隔的字节序列。
打开文件：通过文件流对象的构造函数或成员函数 open，无论哪种都有两个形参。
① 第一个形参是表示文件名的字符串，通常用字符串的起始地址来表示；
② 第二个形参是打开方式，例如 ios::in（只读）、ios::（只写）、ios::app（添加）。
ifstream 构造函数默认只读，ofstream 构造函数默认只写，fstream 构造函数默认读写。

以只写方式打开文件时，如果找不到该文件名的文件，则创建一个该文件名的文件；如果找到该文件名的文件，则会将文件中原有的内容清空。

打开一个文件后，文件流对象就与该文件关联起来，即建立内存与硬盘之间的通道。

关闭一个文件后（使用成员函数 close），文件流对象与该文件之间的关联就切断了。

6、文件的读写

文件的顺序读写和控制台的读写类似。

文件的随机读写先用成员函数 seekg 和 seekp 赋值定位指针，然后从该位置开始读写。

seekg 和 seekp 都有两个参数，第一个用 long 类型的整数表示偏移量，第二个说明该偏移量相对于哪个位置（如 ios::beg、ios::cur、ios::end）。当前位置可以用 tellg 和 tellp 获取。

seekg 和 seekp 通常用于二进制文件的读写，因为二进制文件中每个字段占用的字节数等于该字段的数据在内存中占用的字节数，即只取决于数据类型而与数值无关。在此情况下，可以根据记录的序号以及记录的数据组成结构，来确定该记录（及其内部字段）在二进制文件中的具体位置。

读写二进制文件可以用成员函数 read 和 write 进行无格式的输入和输出。

由于无格式的输入输出不根据数据类型做额外处理，只做字节的传递，因此速度很快。

read 和 write 的第一个参数都是字符指针，表示读/写的起始位置；第二个参数都是整型数，表示读/写的字节个数。

【看看是否完全理解了下面两条语句执行的操作】

```
fout.write ( reinterpret_cast<char *> (&x), sizeof(x) );
fin.read ( reinterpret_cast<char *> (&x), sizeof(x) );
```

表达式 reinterpret_cast<B> a 将数据 a 强制转换为 B 类型。

关于 C++强制类型转换的更多知识参见 https://c.biancheng.net/view/410.html