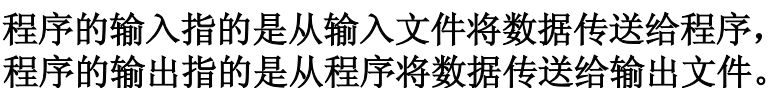
流输入/输出

input：简单来说，从输入设备到主内存

output：从主存储设备到输出设备

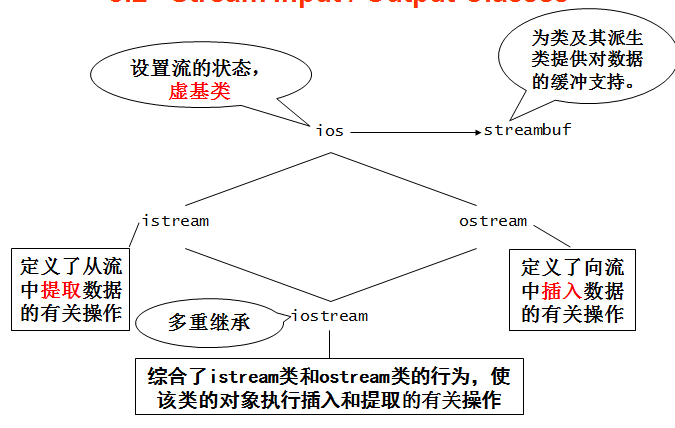


C++的输入与输出包括以下3方面的内容:

（1） 对系统指定的标准设备的输入和输出。

（2） 以外存磁盘文件为对象进行输入和输出，即从磁盘文件输入数据，数据输出到磁盘文件。

（3） 对内存中指定的空间进行输入和输出。



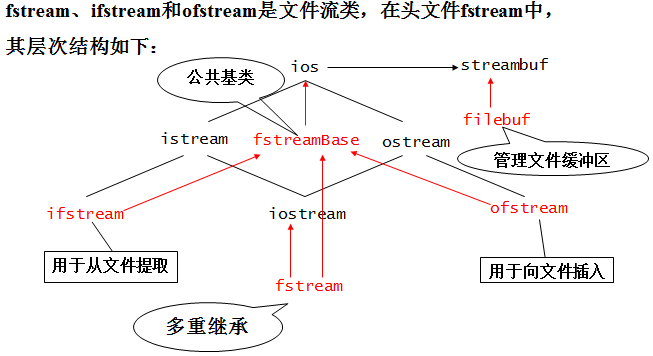
【就看懂了ios与缓冲区相关联，istream和ostream 继承虚基类ios，一个是输出一个是输出，并且iostream多重继承以上两个，可以执行输入和输出】

Stream Input / Output Classes

输入输出流被定义为类，C++的I/O库中的类称为流类(stream class)，用流类定义的对象称

为流对象。cin和cout是iostream的对象。

文件流类



【其实意思就是它的继承方式跟输入输出流差不多的意思？】

大体结构是一样的。

面向对象的标准库

标准库使用了继承来定义一组面向对象类。IO类型在三个独立的头文件中定义：

* iostream定义读写控制窗口的类型
* fstream定义读写已命名文件的类型
* sstream定义了读写内存中string对象的类型

fstream和sstream都里面定义的每种类型都是从iostream中继承(重写)

【就是上面的三个都是iostream类库的，但是作用不完全相同】

IO对象不可复制或赋值

* + 不能把流对象存储在vector等容器中：因为只有支持复制的元素类型才可以存储在vector等容器中，而流对象不支持复制
  + 形参或返回类型不能为流类型：如果需要传递或返回IO对象，必须传递或返回指向该对象的指针或引用

【看懂了原因，但什么叫“把对象存储在容器中”？？】

你需要去了解什么是容器

C++里面的容器，常用的有：vector、list、set、map

容器是用来存放对象的，就像你开一个数组，就是用来存放对应类型的对象，比如vector<int> v; 这个v（容器）就是用来存放整型变量（对象）的。

* 一般情况下，如果需要传递IO对象对它进行读写，则必须使用非const引用，因为对IO对象的读写会改变它的状态



【前面错误的例子懂了，就是如果这样写会涉及到拷贝构造函数，而流对象的拷贝构造是私有的。然后后面正确的部分，可以使用引用。是因为引用的话就不会涉及到拷贝构造，而是直接复制的是指向的内存（？）】

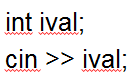
使用引用的话，这两个变量名虽然不同，但是在内存里面是相同的空间，它们是完全相同的对象。

输入输出操作符

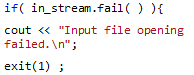
支持级联(Cascading)

可以重载

条件状态

如 如果输入的是int就是成功读取，为无错误状态，如果是其他类型，虽然也可以输入（进行了类型转换），但产生一个错误状态

检测的最简单方法是用if



【这个也算条件状态的判断吧？】

这里是根据是否产生错误状态来做出反应，如果产生了错误状态，则先输入一句话，然后程序退出。

为什么IO对象（如上面的cin）可以直接用于条件判断：

IO对象都重载了转换操作符operator bool()，在用于条件判断的时候，因为需要一个bool类型，编译器就会对IO对象进行转换，当IO对象的条件不处于任何一种错误状态的时候就会返回true。

输出缓冲区的管理

* 每个IO对象管理一个输出缓冲区，用于存储程序读写的数据，如语句：

cout << “please enter a value”;

系统将字符串字面值存储在cout的缓冲区中，并没有输出到设备或者文件中，如上面的语句并没有马上显示在控制窗口中。

缓冲区被刷新的时候，缓冲区中的内容会被写入真实的输出设备或者文件中

【为什么黑框输出的时候没有endl也是立刻输出的？】

因为缓冲区刷新了…实际上在老式的iostream库中，就可以发现你想要得到的结果，没有endl和其他刷新缓冲区的操作符的时候它没有立即被输出。

输出缓冲区的刷新

* 程序正常结束
* 缓冲区已经满了。在这种情况下，缓冲区将会在写下一个值之前刷新
* 用操纵符显式地刷新缓冲区，例如endl（用于输出一个换行符并刷新缓冲区）
* 使用unitbuf操作符设置流内部状态
* 将输出流与输入流关联(tie)起来。在这种情况下，在读输入流时将刷新其关联的输出缓冲区【就是cin的时候cout缓冲区会刷新，然后cout的会输出？？】

操纵符

* endl：用于输出一个换行符并刷新缓冲区
* flush：用于刷新流，但不在输出中添加任何字符【就是直接输出吧……】
* ends：在缓冲区中插入空字符null，并刷新缓冲区【空字符？应该不是空格……就是什么都没有，跟上面有区别么？】
* 最简单的解释：一个是有字符的，一个是没有字符的，有字符自然要占用空间了…
* 表面上看起来是没有区别的，空字符是一个字符，就是字符串最后的’\0’，这是要占空间的。
* unitbuf操纵符：需要刷新所有输出的时候使用，若要取消unitbuf的作用可以使用nounitbuf操纵符，它将流恢复为使用正常的、由系统管理的缓冲区刷新方式

【关于操纵符这部分，上面的东西大概能看懂，不过其实没搞懂那个缓冲区是怎么回事。】

搞不懂就先忽略吧，其实我也不懂…想搞懂的话除非去把C++标准库的相关源码给读了…

不过暂时没这个必要。

文件流对象的使用

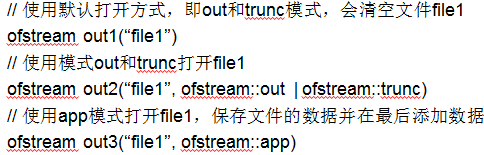
fstream头文件中定义了三种支持文件IO的类型：

* ifstream，由istream派生而来，提供读文件功能
* ofstream，由ostream派生而来，提供写文件功能
* fstream，由iostream派生而来，提供读写同一个文件的功能

一般分开比较好，同时读写会比较容易出问题。

例子

【或者txt文件。两种格式使用上应该没有区别。】



【就是out1和out2是等价的，然后out3是数据不断增加，我的输入更改了重新输出，原本的数据没有被清空，而是在后面不断增加。】

stringstream特定的操作

这个的对象是string类，【没有尝试过使用】

格式状态

就是用操作符使输出符合我们想要的格式，例如

* + 输出元素的宽度
  + 浮点数的的格式，如精度、记数法等
  + 整型值的基数，如十进制、十六进制等
  + 其他一些格式化特征

【具体使用待尝试】