1 IO类

参考文档：IO库类型和头文件

iostream定义了用于读写流的基本类型，fstream定义了读写命名文件类型，sstream定义了读写内存string对象的类型

为了支持使用宽字符的语言，标准库定义了一组类型和对象来操纵wchar\_r类型的数据。宽字符版本的类型和函数的名字以一个w开始。

类型ifstream和istringstream都继承自istream。

不能拷贝或对IO对象赋值。

IO操作一个与生俱来的问题是可能发生错误，一些错误是可恢复的，其他错误则发生在系统深处。

参考文档：IO库条件状态

4种IO库条件状态：strm::badbit,failbit,eofbit,goodbit

确定一个流的状态的最简单的方法是将它当作一个条件来使用while(cin >> word)。

IO库定义了一个与机器无关的iostate类型，它提供了表达流状态的完整功能。IO库定义了4个iostate类型的constexpr值表示特定的位模式。

badbit表示系统级错误，failbit表示可恢复错误，比如期望读取数值却读出一个字符等错误。如果到达文件结束位置，eofbit和failbit都会被置位。

goodbit的值为0，表示流未发生错误。如果badbit,failbit和eofbit任一个被置位，则检测流状态的条件会失败。

good在所有错误位均未置位的情况下返回true，而bad,fail和eof则在对应错误位被置位时返回true。此外，在badbit被置位时，fail也会返回true。

使用good和fail是确定流的总体状态的正确做法。而eof和bad操作只能表示特定的错误。

流对象的rdstate成员返回一个iostate值，对应流的当前状态。setstate操作将给定条件位置位，表示发生了对应错误。

clear是一个重载函数，不接受参数的版本清楚所有错误标志位，执行后，调用good会返回true。

将failbit和badbit复位，但保持eofbit不变：cin.clear(cin.rdstate() & ~cin.failbit & ~cin.badbit);

每个输出流都管理一个缓冲区，文本串可能立即打印出来，但也可能被操作系统保存在缓冲区中，随后再打印。

参考文档：导致缓冲刷新的原因

除了endl，flush和ends都可以刷新缓冲区。ends向缓冲区插入一个空字符。

如果想在每次输出操作后都刷新缓冲区可以使用unitbuf操纵符。cout << unitbuf;回到正常缓冲方式 cout << nounitbuf;

当一个输入流被关联到一个输出流时，任何视图从输入流读取数据的操作都会先刷新关联的输出流。标准库将cout和cin关联在一起。

因此cin >> ival;会导致cout缓冲区刷新。

将cin和cout关联起来cin.tie(&cout)

2 文件输入输出

ifstream从一个给定文件读取数据，ofstream向一个给定文件写入数据，fstream可以读写给定文件。

参考文档：fstream特有的操作

创建文件流对象时，我们可以提供文件名（可选的）。如果提供了一个文件名，则open会自动被调用。

可以定义一个空文件流对象，然后调用open将它与文件关联起来。

调用open可能会失败，进行open是否成功的检测通常是一个好习惯。

当一个fstream对象被销毁时，close会自动调用。

每个流都有一个关联的文件模式，用来指出如何使用文件。

参考文档：文件模式

默认情况下，当我们打开一个ofstream时，文件的内容会被丢弃，阻止一个ofstream清空给定文件内容的方法是同时指定app模式。

例子：ofstream out("file1", ofstream::out | ofstream::trunc)

3 string流

istringstream从string读取数据，ostringstream向string写入数据，stringstream既可以从string读取数据也可以向string写入数据。

参考文档：stringstream特有的操作

编程：使用istringstream读取名字和电话

编程：使用ostringstream