第8章 IO库

8.1 IO 类

1. `iostream` 定义了用于读写流的基本类型。 `fstream` 定义了读写命名文件的类型。 `sstream` 定义了读写内存 `string` 对象的类型。
2. 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作 `wchar_t` 类型的数据。宽字符版本的类型和函数的名字以 `w` 开始，例如 `wcin`, `wcout` 等。

8.1.1 IO 对象无拷贝或赋值

1. 不能将形参或返回类型设置为流类型。
2. 进行 IO 操作的函数通常以引用方式传递和返回流。
3. 传递和返回的引用不能是 `const` 的。

8.1.2 条件状态

1. 由于流可能处于错误状态，因此代码通常应该在使用一个流之前检查它是处于良好状态，最简单的方法是将它当作一个 *条件* 来使用。
2. 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。这个类型应作为一个 *位集合* 来使用。
3. IO 库定义了 4 个 `iostate` 类型的 `constexpr` 值（`strm::badbit`, `strm::failbit`, `strm::eofbit`, `strm::goodbit`），表示特定的位模式。这些值用来表示特定类型的 IO 条件，可以与位运算符一起使用来一次性检测或设置多个标志位。
4. `badbit` 表示系统级错误，如不可恢复的读写错误。通常情况下一旦 `badbit` 被置位，流就无法再使用了。
5. 在发生可恢复错误后 `failbit` 被置位，如期望读取数值却读出一个字符等错误。这中问题通常是可以修正的，流还可以继续使用。
6. 如果到达文结束位置， `eofbit` 和 `failbit` 都会被置位。
7. `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。
8. 流对象的 `rdstate` 成员返回一个 `iostate` 值，对应流的当前状态。
9. `setstate` 操作将给定条件位置位，表示发生了对应错误。
10. `clear` 不接受参数的版本清除（复位）所有错误标识位。带参数的版本接受一个 `iostate` 值，表示流的新状态。

8.1.3 管理输出缓冲

1. 每一个输出流都管理一个缓冲区，用来保存程序读写的数据。
2. `endl` 完成换行并刷新缓冲区的工作； `flush` 刷新缓冲区但不输出任何额外的字符； `ends` 向缓冲区插入一个空字符，然后刷新缓冲区。
3. 使用 `unitbuf` 告诉流在接下来的每次写操作之后都进行一次 `flush` 操作（每次输出操作后都刷新缓冲区）；使用 `nounitbuf` 则重置流使其恢复使用正常的系统管理的缓冲区刷新机制。
4. 如果程序异常终止，输出缓冲区是不会被刷新的。当调试崩溃程序时实际上代码已经执行了，只是程序崩溃后缓冲区没有被刷新，输出数据被挂起没有打印而已。
5. 当一个输入流被关联到一个输出流时，任何试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。
6. 每个流同时最多只能关联到一个流，但多个流可以同时关联到同一个 `ostream` 。

8.2 文件输入输出

8.2.1 使用文件流对象

1. 当一个 `fstream` 对象被销毁时， `close` 会自动被调用。

8.2.2 文件模式

1. 保留 `ofstream` 打开的文件中已有数据的唯一方法是显示指定 `app` 或 `in` 模式。