|  |  |
| --- | --- |
| 1. 流随机访问 2. Seek和tell函数 3. 只有一个标记 4. 重定位标记 5. 访问标记 6. 读写同一个文件 | Seek tell  Seekg(new\_position)  Seekp(offset,from)  Ostringstream::pos\_type mark = writestr.tellp();  Inout.seekp(0,fstream::end) |
| 1. 流随机访问 2. 随机IO本质上是依赖于系统，需查询文档系统 3. 所有流类型都定义了tell seek函数，但是适用于fstream和sstream 4. Seek和tell函数 5. 两对seek和tell，g p 6. Seek是跳，tell是标记。In.Seekp(0,fstream::end),auto end = inout.tellg() 7. 输入与输出版本差别在于名字后缀是g还是p，g表示获得读取，p表示放置写入 8. 对应ifstream使用 g，ofstream使用p,fstream都可以用 9. 只有一个标记 10. 在一个流中只维护单一的标记，不存在独立的读标记和写标记 11. Fstream和stringstream类型可以读写同一个流（就是fstream具有对流既可以读也可以写）。这这些类型中，有单一的缓冲区用于保存读写的数据，标记也只有一个,表示缓冲区中的当前位置。标准库将g和p版本的读写位置都映射到这个单一的标记。   Note：由于只有单一标记，只要我们在读写操作间切换，必须进行seek操作来重定位标记，因为读与写的位置不一样，一般来说读在流中间部分，写在流末尾部分   1. 重定位标记 2. Seek函数两个版本共4个重载，一个移动到文件的绝对位置，一个移动到给定位置的指定偏移量 3. New\_position offset的类型分布是pos\_type和off\_type，与机器相关的，off\_type可以负，向前向后移动 4. 访问标记 5. Tellp teeeg返回一个pos\_type值，表示一个位置 6. 读写同一个文件 7. 看例子即可，主要的实现 8. 用end\_mark = inout.tellg();//记住原文件尾位置，判断是否读完 9. 循环体先记录读取当前位置的mark中，再将写定位到文件尾进行写入，再通过seekg恢复到读取的位置 10. 记住读取的位置mark是每一行的第一个位置，end\_mark也是最后一行的下一行首位置，这样来匹配是否读取完 | |