|  |  |
| --- | --- |
| 1. 未格式化的输入输出操作 2. 单字节操作 3. 将字符放回输入流 4. 从输入操作返回int值 5. 多字节操作 6. 确定读取了多少个字符 | 格式化IO  未格式化IO  Cin.get(ch)  Int ch;  Ch = cin.get() != EOF  Cout.put(ch)  Is.gcount(); |
| 1. 未格式化的输入输出操作 2. 上一节的格式化陈为 格式化IO 3. 一组底层操作，支持未格式化IO。允许将一个流当做一个无解释的字节序列来处理 4. 单字节操作 5. Get 与put 6. 将字符放回输入流 7. 将字符放回流中，这是初衷 8. Peek与get类似，读取下一个字符的副本，但是peek不会删除它 9. Unget，读取下一个字符并且将其放入字符流的末尾 10. Putback 与unget的操作一样，但是带有参数，表示只与这个参数读取的值相同才放入字符流的末尾 11. 从输入操作返回int值 12. Get peek都是以int类型从输入流返回一个字符 13. 因为文件尾标记是int，一个负值，可以判断是否到达末尾 14. 先将返回的字符转换为unsigned char 再将结果提升到int，即使字符集中有字符映射为负值也会转换为无符号的unsigned char再到int，正值与负值确定是否到达文件尾 15. 多字节操作 16. 未格式化的IO操作一次处理大块数据，因为快。容易出错，因为自己分配并管理用来保存和提取数据的字符数组 17. Sink是字符数组的起始地址。 18. Get getline区别：getline会读取并且丢弃delim字符，get是直至delim不读取并结束   并且get将分隔符留作istream的下一个字符  Getline是读取并且丢弃分隔符  但都不会保留分隔符在sink中   1. 确定读取了多少个字符 2. 从输入操作读取未知个数的字节。可以调用gcount来确定最后一个未格式化输入操作读取了多少个字符 3. 应该在任何后续未格式化输入操作之前调用gcount 4. 字符退回流的单字符操作是未格式化操作，则gcount之前用了peek、unget putback 则gcount返回0，因为它只是退回没有读取   Note：底层函数容易出错。   1. 若get peek返回值给予一个char 2. 若一台char实现为unsigned char机器上，char会为正值永远不等于负值 3. 若一台char实现为signed char机器上，\377字符对应-1的值，则可能提前退出   读写有类型的值时，错误不会发生，应使用标准库提供的类型 | |





