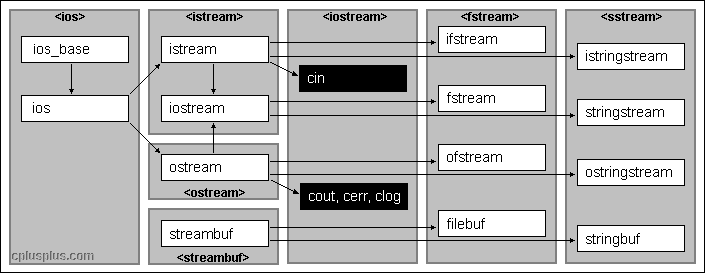
**POCO C++库学习和分析 -- 流 （一）**

          流(Stream)是C++和C之间的一大区别。写C++的程序员都知道流的用法。在Poco库中，在标准流的基础上又扩充了一些流，分别是基于Base64和HexBinary的编解码流，使用zlib的数据压缩流，二进制的I/O流，文件流，以及一些其他的辅助流；另外Poco库还提供了一个扩展的结构，用于创建用户自定义流。  
  
          Poco库中所有的流类都与标准c++库中的流兼容。并且在Poco库中，大多数流都仅仅是个过滤器，这意味着它们不会直接从设备中读取或者写入数据，通常情况下它们会链接到另一个流上。下面我们分别对它们进行介绍。

**1. 标准c++流介绍**

          在介绍Poco的流之前，我觉得有必要了解C++中的输入输出流，不然就会觉得Poco中的流很难理解。在看完C++的流结构后，自然会对Poco库中的流内容豁然开朗。我也一样。  
          为了保证语言和平台无关，C++和C一样，不具备内部输入输出能力。语言的输入输出能力是和操作系统相关的，在最底层都是通过调用操作系统的I/O库实现。  
  
          在C++的iostream流库中，存在着两个基本部分。分别是：  
          1. 流：C++把输入和输出看作字节流。输入时，程序从输出流中抽取字节；输出时，程序将字节插入到输出流中。流充当了程序和流源或者流目标之间的桥梁。  
          2. 缓冲区：缓冲区是用作中介的内存块，它是将信息从设备传输到程序或者从程序传输到设备的临时存储工具，用以匹配程序和设备之间速度的差距。从设计上说，增加了缓冲区，使的C++的iostream结构更具有扩展性。  
  
          C++的[输入输出类图](http://www.cplusplus.com/img/iostream.gif)：



          下面对C++中各个流类的介绍主要来自于wiki以及网站[cplusplus](http://www.cplusplus.com/)。

**1.1 ios\_base**

          ios\_base类封装了C++标准中的流，并定义了在输入输出中不依赖于读写的数据类型的基本信息和行为，如格式化信息、异常状态、事件回调等。  
  
          在类std::ios\_base中，保存了下述关于流的信息：  
          格式控制信息的枚举类型fmtflags ，影响到如何解释输入串行的格式、如何生成输出串行的格式，例如整数是用16进制还是10进制表示，浮点数是科学计数法还是定点形式；  
          流的状态枚举类型iostate，如数据是否完整、是否到达流的末尾、是否读写失败等；  
          流的打开方式枚举类型openmode，如读取、写入、追加、创建时删除原内容、二进制打开、  
          流的定位位置枚举类型seekdir，如开始位置、当前位置、结尾位置等。  
          流的事件枚举类型event，如“擦除”事件erase\_event，改变locale设置事件imbue\_event，复制格式事件copyfmt\_event。  
          流的私有的其它额外保存的数据，为一个long型数组与一个指针数组。  
          一个成员类failure，用于作为C++标准中，流输入输出类库抛出的各种异常的基类。  
          一个成员类Init，用于封装cout、cin、wcout等8个静态对象的初始化函数。  
  
**成员函数包括：  
  
          *格式化:***

          1. 读取/设置流的格式

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. fmtflags flags() **const**;
2. fmtflags flags (fmtflags fmtfl);

  例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // modify flags
2. #include <iostream>     // std::cout, std::ios
4. **int** main () {
5. std::cout.flags ( std::ios::right | std::ios::hex | std::ios::showbase );
6. std::cout.width (10);
7. std::cout << 100 << '\n';
8. **return** 0;
9. }

          2. 设置流的格式，与原有格式合并

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. fmtflags setf (fmtflags fmtfl);
2. fmtflags setf (fmtflags fmtfl, fmtflags mask);

  例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // modifying flags with setf/unsetf
2. #include <iostream>     // std::cout, std::ios
4. **int** main () {
5. std::cout.setf ( std::ios::hex, std::ios::basefield );  // set hex as the basefield
6. std::cout.setf ( std::ios::showbase );                  // activate showbase
7. std::cout << 100 << '\n';
8. std::cout.unsetf ( std::ios::showbase );                // deactivate showbase
9. std::cout << 100 << '\n';
10. **return** 0;
11. }

输出:  
         Output:  
         0x64  
         64  
  
  
          3. 根据参数mask，清除流的格式的某些位(bit)

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **void** unsetf (fmtflags mask);

  例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // modifying flags with setf/unsetf
2. #include <iostream>     // std::cout, std::ios
4. **int** main () {
5. std::cout.setf ( std::ios::hex, std::ios::basefield );  // set hex as the basefield
6. std::cout.setf ( std::ios::showbase );                  // activate showbase
7. std::cout << 100 << '\n';
8. std::cout.unsetf ( std::ios::showbase );                // deactivate showbase
9. std::cout << 100 << '\n';
10. **return** 0;
11. }

输出:  
         0x64  
         64  
  
  
          4. 读取/设置显示浮点数时的精度

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streamsize precision() **const**;
2. streamsize precision (streamsize prec);

  例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // modify precision
2. #include <iostream>     // std::cout, std::ios
4. **int** main () {
5. **double** f = 3.14159;
6. std::cout.unsetf ( std::ios::floatfield );                // floatfield not set
7. std::cout.precision(5);
8. std::cout << f << '\n';
9. std::cout.precision(10);
10. std::cout << f << '\n';
11. std::cout.setf( std::ios::fixed, std:: ios::floatfield ); // floatfield set to fixed
12. std::cout << f << '\n';
13. **return** 0;
14. }

输出:  
         3.1416  
         3.14159  
         3.1415900000  
  
  
          5. 读取/设定流的输出数据的显示宽度

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streamsize width() **const**;
2. streamsize width (streamsize wide);

  例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // field width
2. #include <iostream>     // std::cout, std::left
4. **int** main () {
5. std::cout << 100 << '\n';
6. std::cout.width(10);
7. std::cout << 100 << '\n';
8. std::cout.fill('x');
9. std::cout.width(15);
10. std::cout << std::left << 100 << '\n';
11. **return** 0;
12. }

输出:  
         100  
                100  
         100xxxxxxxxxxxx  
  
  
          ***语言环境:***  
          1. 给流设置本地语言环境

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. locale imbue (**const** locale& loc);

  例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // imbue example
2. #include <iostream>     // std::cout
3. #include <locale>       // std::locale
5. **int** main()
6. {
7. std::locale mylocale("");   // get global locale
8. std::cout.imbue(mylocale);  // imbue global locale
9. std::cout << 3.14159 << '\n';
10. **return** 0;
11. }

输出:

         3,14159

          2. 获取当前使用语言环境

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. locale getloc() **const**;

**1.2 basic\_ios**

          basic\_ios定义出“与字符类型及其相应字符特性相关”的stream class的共同属性，其中包括清除流状态、设置流状态、拷贝流标志、返回或设置流缓冲区指针、设置本地化相关信息、返回或设置填充字符、字符转换，还包括了stream所用的缓冲器.  
          basic\_ios在其内部定义了一个指向streambuf的指针。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **template** <**class** Elem, **class** Traits>
2. **class** basic\_ios
3. : **public** ios\_base
4. {
5. //C++标准库封装了一个缓冲区类streambuf，以供输入输出流对象使用。
6. //每个标准C++输出输出流对象都包含一个指向streambuf的指针，
7. basic\_streambuf<\_Elem, \_Traits>\*\_Mystrbuf;

10. // ....
11. }

         **成员函数包括:**  
  
         1. 状态标记函数:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **bool** good() **const**; //检查流状态位是否为good
2. **bool** eof() **const**; //检查流状态位是否为eof，eofbit位被标志
3. **bool** fail() **const**; //检查流状态位是否为fail，failbit或者badbit被标志
4. **bool** bad() **const**; //检查流状态位是否为bad，badbit位被标志
5. iostate rdstate() **const**; //返回流状态位

          有两种方法可以获得输入/输出的状态信息。一种方法是通过调用rdstate()函数，它返回当前状态的错误标记。另一种方法则是使用good(), eof(),fail(), bad()函数来检测相应的输入/输出状态。  
          状态位和函数返回值关系如下表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| [**iostate**](http://www.cplusplus.com/ios_base::iostate)**value (member constant)** | **indicates** | **functions to check state flags** | | | | |
| [**good()**](http://www.cplusplus.com/ios::good) | [**eof()**](http://www.cplusplus.com/ios::eof) | [**fail()**](http://www.cplusplus.com/ios::fail) | [**bad()**](http://www.cplusplus.com/ios::bad) | [**rdstate()**](http://www.cplusplus.com/ios::rdstate) |
| **goodbit** | No errors (zero value [iostate](http://www.cplusplus.com/ios_base::iostate)) | true | false | false | false | goodbit |
| **eofbit** | End-of-File reached on input operation | false | true | false | false | eofbit |
| **failbit** | Logical error on i/o operation | false | false | true | false | failbit |
| **badbit** | Read/writing error on i/o operation | false | false | true | true | badbit |

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // error state flags
2. #include <iostream>     // std::cout, std::ios
3. #include <sstream>      // std::stringstream
5. **void** print\_state (**const** std::ios& stream) {
6. std::cout << " good()=" << stream.good();
7. std::cout << " eof()=" << stream.eof();
8. std::cout << " fail()=" << stream.fail();
9. std::cout << " bad()=" << stream.bad();
10. }
12. **int** main () {
13. std::stringstream stream;
15. stream.clear (stream.goodbit);
16. std::cout << "goodbit:"; print\_state(stream); std::cout << '\n';
18. stream.clear (stream.eofbit);
19. std::cout << " eofbit:"; print\_state(stream); std::cout << '\n';
21. stream.clear (stream.failbit);
22. std::cout << "failbit:"; print\_state(stream); std::cout << '\n';
24. stream.clear (stream.badbit);
25. std::cout << " badbit:"; print\_state(stream); std::cout << '\n';
27. **return** 0;
28. }

输出:  
         goodbit: good()=1 eof()=0 fail()=0 bad()=0  
         eofbit: good()=0 eof()=1 fail()=0 bad()=0  
         failbit: good()=0 eof()=0 fail()=1 bad()=0  
         badbit: good()=1 eof()=0 fail()=1 bad()=1

         2. oprator!()

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **bool** operator!() **const**;

         如果没有错误标记被设置(failbit或badbit),返回true，否则返回false

         3. 设置/清除状态

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **void** setstate (iostate state);
2. **void** clear (iostate state = goodbit);

         有两种方法可以设置输入/输出的状态信息。clear()函数可以使流状态将按照ios\_base::iostate所描述的样子进行设置。ios::failbit、ios::badbit、ios::eofbit、ios::goodbit均为常量，它们中的任何一个都代表了一种流状态，或可被称为“输入状态标记位常量”。它们不是failbit、badbit、eofbit、goodbit这四个标记位的存贮变量。标记为常量的状态如上表所述。  
  
         clear() 函数作用是：将流状态设置成括号内参数所代表的状态，强制覆盖掉流的原状态。

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // clearing errors
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::fstream
5. **int** main () {
6. **char** buffer [80];
7. std::fstream myfile;
9. myfile.open ("test.txt",std::fstream::in);
11. myfile << "test";
12. **if** (myfile.fail())
13. {
14. std::cout << "Error writing to test.txt\n";
15. myfile.clear();
16. }
18. myfile.getline (buffer,80);
19. std::cout << buffer << " successfully read from file.\n";
21. **return** 0;
22. }

         setstate()函数的作用是：它并不强制覆盖流的原状态，而是将括号内参数所代表的状态叠加到原始状态上。它相当于：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **void** basic\_ios::setstate (iostate state) {
2. clear(rdstate()|state);
3. }

         4. 拷贝格式:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. basic\_ios& copyfmt (**const** basic\_ios& rhs);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // copying formatting information
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ofstream
5. **int** main () {
6. std::ofstream filestr;
7. filestr.open ("test.txt");
9. std::cout.fill ('\*');
10. std::cout.width (10);
11. filestr.copyfmt (std::cout);
13. std::cout << 40;
14. filestr << 40;
16. **return** 0;
17. }

         5. 设置或获取填充字符

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. char\_type fill() **const**;
2. char\_type fill (char\_type fillch);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // using the fill character
2. #include <iostream>     // std::cout
4. **int** main () {
5. **char** prev;
7. std::cout.width (10);
8. std::cout << 40 << '\n';
10. prev = std::cout.fill ('x');
11. std::cout.width (10);
12. std::cout << 40 << '\n';
14. std::cout.fill(prev);
16. **return** 0;
17. }

输出:  
                 40  
         xxxxxxxx40

         6. 返回和设置当前流的异常标志

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. iostate exceptions() **const**;
2. **void** exceptions (iostate except);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // basic\_ios::exceptions
2. #include <iostream>     // std::cerr
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. std::ifstream file;
7. file.exceptions ( std::ifstream::failbit | std::ifstream::badbit );
8. **try** {
9. file.open ("test.txt");
10. **while** (!file.eof()) file.get();
11. }
12. **catch** (std::ifstream::failure e) {
13. std::cerr << "Exception opening/reading file";
14. }
16. file.close();
18. **return** 0;
19. }

         7. 返回和设置绑定的输出流(绑定流是一个输出流，输出流将在流的每一次I/O操作前先被输出)。“绑定”的效果也就相当于，每当被“绑定”的对象有输入或输出操作时，会先自动刷新“绑定”对象的缓冲区，以达到实时的效果。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. basic\_ostream<char\_type,traits\_type>\* tie() **const**;
2. basic\_ostream<char\_type,traits\_type>\* tie (basic\_ostream<char\_type,traits\_type>\* tiestr);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // redefine tied object
2. #include <iostream>     // std::ostream, std::cout, std::cin
3. #include <fstream>      // std::ofstream
5. **int** main () {
6. std::ostream \*prevstr;
7. std::ofstream ofs;
8. ofs.open ("test.txt");
10. std::cout << "tie example:\n";
12. \*std::cin.tie() << "This is inserted into cout";
13. prevstr = std::cin.tie (&ofs);
14. \*std::cin.tie() << "This is inserted into the file";
15. std::cin.tie (prevstr);
17. ofs.close();
19. **return** 0;
20. }

输出:  
         tie example:  
         This is inserted into cout  
  
  
         8. 返回或设置流缓冲区指针。用户可以通过调用rdbuf()成员函数获得该指针，从而直接访问底层streambuf对象。因此，可以直接对底层缓冲区进行数据读写，从而跳过上层的格式化输入输出操作。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. basic\_streambuf<char\_type,traits\_type>\* rdbuf() **const**;
2. basic\_streambuf<char\_type,traits\_type>\* rdbuf (basic\_streambuf<char\_type,traits\_type>\* sb);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // redirecting cout's output thrrough its stream buffer
2. #include <iostream>     // std::streambuf, std::cout
3. #include <fstream>      // std::ofstream
5. **int** main () {
6. std::streambuf \*psbuf, \*backup;
7. std::ofstream filestr;
8. filestr.open ("test.txt");
10. backup = std::cout.rdbuf();     // back up cout's streambuf
12. psbuf = filestr.rdbuf();        // get file's streambuf
13. std::cout.rdbuf(psbuf);         // assign streambuf to cout
15. std::cout << "This is written to the file";
17. std::cout.rdbuf(backup);        // restore cout's original streambuf
19. filestr.close();
21. **return** 0;
22. }

         9. 字符转换  
         返回宽字符wc对应的字符dfault，在其内部使用了流对应的locale对象做转换

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **char** narrow (char\_type wc, **char** dfault) **const**;

         返回c对应的宽字符

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. char\_type widen (**char** c) **const**;

**1.3  输入流（ istream）**

         istream是普通输入流类和用于其它输入流的基类。  
  
        **输入流的成员函数包括：**  
  
         1. 构造函数

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **explicit** istream (streambuf\* sb);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream constructor
2. #include <iostream>     // std::ios, std::istream, std::cout
3. #include <fstream>      // std::filebuf

6. **int** main () {
7. std::filebuf fb;
8. **if** (fb.open ("test.txt",std::ios::in))
9. {
10. std::istream is(&fb);
11. **while** (is)
12. std::cout << **char**(is.get());
13. fb.close();
14. }
15. **return** 0;
16. }

         2. 重载析取运算符

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& operator>> (**bool**& val);
2. istream& operator>> (**short**& val);
3. istream& operator>> (unsigned **short**& val);
4. istream& operator>> (**int**& val);
5. istream& operator>> (unsigned **int**& val);
6. istream& operator>> (**long**& val);
7. istream& operator>> (unsigned **long**& val);
8. istream& operator>> (**float**& val);
9. istream& operator>> (**double**& val);
10. istream& operator>> (**long** **double**& val);
11. istream& operator>> (**void**\*& val);
12. stream buffers (2)
13. istream& operator>> (streambuf\* sb );
14. manipulators (3)
15. istream& operator>> (istream& (\*pf)(istream&));
16. istream& operator>> (ios& (\*pf)(ios&));
17. istream& operator>> (ios\_base& (\*pf)(ios\_base&));

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // example on extraction
2. #include <iostream>     // std::cin, std::cout, std::hex
4. **int** main () {
5. **int** n;
7. std::cout << "Enter a number: ";
8. std::cin >> n;
9. std::cout << "You have entered: " << n << '\n';
11. std::cout << "Enter a hexadecimal number: ";
12. std::cin >> std::hex >> n;         // manipulator
13. std::cout << "Its decimal equivalent is: " << n << '\n';
15. **return** 0;
16. }

         3. 无格式输入：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streamsize gcount() **const**;

         获取最后一次执行析取操作所读取的字符串数目

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // cin.gcount example
2. #include <iostream>     // std::cin, std::cout
4. **int** main () {
5. **char** str[20];
7. std::cout << "Please, enter a word: ";
8. std::cin.getline(str,20);
9. std::cout << std::cin.gcount() << " characters read: " << str << '\n';
11. **return** 0;
12. }

输出:  
         Please, enter a word: simplify  
         9 characteres read: simplify  
  
  
         4. 从流中以无格式输入方式析取字符串

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** get()
2. istream& get (**char**& c)
3. istream& get (**char**\* s, streamsize n)
4. istream& get (**char**\* s, streamsize n, **char** delim)
5. istream& get (streambuf& sb)
6. istream& get (streambuf& sb, **char** delim)

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::get example
2. #include <iostream>     // std::cin, std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. **char** str[256];
8. std::cout << "Enter the name of an existing text file: ";
9. std::cin.get (str,256);    // get c-string
11. std::ifstream is(str);     // open file
13. **while** (is.good())          // loop while extraction from file is possible
14. {
15. **char** c = is.get();       // get character from file
16. **if** (is.good())
17. std::cout << c;
18. }
20. is.close();                // close file
22. **return** 0;
23. }

         5. 从流中以无格式输入方式析取字符串(直至提取到指定的分隔字符，或已向流中写入最大长度的n个字符(包括终止空字符))，并存储内容进入c风格的字符串中。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& getline (**char**\* s, streamsize n );
2. istream& getline (**char**\* s, streamsize n, **char** delim );

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::getline example
2. #include <iostream>     // std::cin, std::cout
4. **int** main () {
5. **char** name[256], title[256];
7. std::cout << "Please, enter your name: ";
8. std::cin.getline (name,256);
10. std::cout << "Please, enter your favourite movie: ";
11. std::cin.getline (title,256);
13. std::cout << name << "'s favourite movie is " << title;
15. **return** 0;
16. }

         6. 从流中提取字符输入序列(直到n个字符被提取，或遇见一个指定的字符分隔符)，并丢弃。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& ignore (streamsize n = 1, **int** delim = EOF);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::ignore example
2. #include <iostream>     // std::cin, std::cout
4. **int** main () {
5. **char** first, last;
7. std::cout << "Please, enter your first name followed by your surname: ";
9. first = std::cin.get();     // get one character
10. std::cin.ignore(256,' ');   // ignore until space
12. last = std::cin.get();      // get one character
14. std::cout << "Your initials are " << first << last << '\n';
16. **return** 0;
17. }

         7. 返回输入序列中的下一个字符，但不析取：字符将在下一次析取操作中从流中提取。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** peek();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::peek example
2. #include <iostream>     // std::cin, std::cout
3. #include <string>       // std::string

6. **int** main () {
8. std::cout << "Please, enter a number or a word: ";
9. **char** c = std::cin.peek();
11. **if** ( (c >= '0') && (c <= '9') )
12. {
13. **int** n;
14. std::cin >> n;
15. std::cout << "You entered the number: " << n << '\n';
16. }
17. **else**
18. {
19. std::string str;
20. std::getline (std::cin, str);
21. std::cout << "You entered the word: " << str << '\n';
22. }
24. **return** 0;
25. }

输出:  
         Please, enter a number or a word: foobar  
         You entered the word: foobar  
  
  
         8. 从流中析取n个字符，并把它们存储到s指向的字符数组

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& read (**char**\* s, streamsize n);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // read a file into memory
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. std::ifstream is ("test.txt", std::ifstream::binary);
7. **if** (is) {
8. // get length of file:
9. is.seekg (0, is.end);
10. **int** length = is.tellg();
11. is.seekg (0, is.beg);
13. **char** \* buffer = **new** **char** [length];
15. std::cout << "Reading " << length << " characters... ";
16. // read data as a block:
17. is.read (buffer,length);
19. **if** (is)
20. std::cout << "all characters read successfully.";
21. **else**
22. std::cout << "error: only " << is.gcount() << " could be read";
23. is.close();
25. // ...buffer contains the entire file...
27. **delete**[] buffer;
28. }
29. **return** 0;
30. }

输出:  
         Reading 640 characters... all characters read successfully.  
  
  
         9. 从流中析取n个字符，并把它们存储到s指向的字符数组。在内部关联的缓冲区(如果存在的话)用完时，操作返回，即使文件结束字符尚未达到。这个函数主要用于从某些类型的异步源中读取数据。由于析取操作会在内部缓冲区耗尽时终止，从而避免了潜在的延时。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streamsize readsome (**char**\* s, streamsize n);

         10. 把指定字符还给输入流，使得字符能够被重新析取

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& putback (**char** c);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::putback example
2. #include <iostream>     // std::cin, std::cout
3. #include <string>       // std::string
5. **int** main () {
6. std::cout << "Please, enter a number or a word: ";
7. **char** c = std::cin.get();
9. **if** ( (c >= '0') && (c <= '9') )
10. {
11. **int** n;
12. std::cin.putback (c);
13. std::cin >> n;
14. std::cout << "You entered a number: " << n << '\n';
15. }
16. **else**
17. {
18. std::string str;
19. std::cin.putback (c);
20. getline (std::cin,str);
21. std::cout << "You entered a word: " << str << '\n';
22. }
23. **return** 0;
24. }

输出:  
         Please, enter a number or a word: pocket  
         You entered a word: pocket  
  
  
         11. 把上一个析取的字符还给输入流，使得字符能够被重新析取

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& unget();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::unget example
2. #include <iostream>     // std::cin, std::cout
3. #include <string>       // std::string
5. **int** main () {
6. std::cout << "Please, enter a number or a word: ";
7. **char** c = std::cin.get();
9. **if** ( (c >= '0') && (c <= '9') )
10. {
11. **int** n;
12. std::cin.unget();
13. std::cin >> n;
14. std::cout << "You entered a number: " << n << '\n';
15. }
16. **else**
17. {
18. std::string str;
19. std::cin.unget();
20. getline (std::cin,str);
21. std::cout << "You entered a word: " << str << '\n';
22. }
23. **return** 0;
24. }

输出:  
         Please, enter a number or a word: 7791  
         You entered a number: 7791  
  
  
         12. 返回当前输入流字符的位置：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streampos tellg();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // read a file into memory
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. std::ifstream is ("test.txt", std::ifstream::binary);
7. **if** (is) {
8. // get length of file:
9. is.seekg (0, is.end);
10. **int** length = is.tellg();
11. is.seekg (0, is.beg);
13. // allocate memory:
14. **char** \* buffer = **new** **char** [length];
16. // read data as a block:
17. is.read (buffer,length);
19. is.close();
21. // print content:
22. std::cout.write (buffer,length);
24. **delete**[] buffer;
25. }
27. **return** 0;
28. }

         13. 设置下一个析取字符串在输入流中的位置

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& seekg (streampos pos);
2. istream& seekg (streamoff off, ios\_base::seekdir way);

**同步**  
         1. 同步流相关的缓冲区buf

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** sync();

  例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // syncing input stream
2. #include <iostream>     // std::cin, std::cout
4. **int** main () {
5. **char** first, second;
7. std::cout << "Please, enter a word: ";
8. first = std::cin.get();
9. std::cin.sync();
11. std::cout << "Please, enter another word: ";
12. second = std::cin.get();
14. std::cout << "The first word began by " << first << '\n';
15. std::cout << "The second word began by " << second << '\n';
17. **return** 0;
18. }

输出:  
         Please, enter a word: test  
         Please enter another word: text  
         The first word began by t  
         The second word began by t

**1.4. 输出流（ostream）**

         ostream是普通输出流类和用于其它输出流类的基类。  
  
         1. 构造函数

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **explicit** ostream (streambuf\* sb);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // ostream constructor
2. #include <iostream>     // std::cout, std::ostream, std::ios
3. #include <fstream>      // std::filebuf
5. **int** main () {
6. std::filebuf fb;
7. fb.open ("test.txt",std::ios::out);
8. std::ostream os(&fb);
9. os << "Test sentence\n";
10. fb.close();
11. **return** 0;
12. }

         2. 重载插入运算符

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ostream& operator<< (**bool** val);
2. ostream& operator<< (**short** val);
3. ostream& operator<< (unsigned **short** val);
4. ostream& operator<< (**int** val);
5. ostream& operator<< (unsigned **int** val);
6. ostream& operator<< (**long** val);
7. ostream& operator<< (unsigned **long** val);
8. ostream& operator<< (**float** val);
9. ostream& operator<< (**double** val);
10. ostream& operator<< (**long** **double** val);
11. ostream& operator<< (**void**\* val);
12. ostream& operator<< (streambuf\* sb );
13. ostream& operator<< (ostream& (\*pf)(ostream&));
14. ostream& operator<< (ios& (\*pf)(ios&));
15. ostream& operator<< (ios\_base& (\*pf)(ios\_base&));

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // example on insertion
2. #include <iostream>     // std::cout, std::right, std::endl
3. #include <iomanip>      // std::setw
5. **int** main () {
6. **int** val = 65;
8. std::cout << std::right;       // right-adjusted (manipulator)
9. std::cout << std::setw(10);    // set width (extended manipulator)
11. std::cout << val << std::endl; // multiple insertions
13. **return** 0;
14. }

输出:  
        65  
  
  
         **无格式输出**  
         1. 把字符c插入流中

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ostream& put (**char** c);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // typewriter
2. #include <iostream>     // std::cin, std::cout
3. #include <fstream>      // std::ofstream
5. **int** main () {
6. std::ofstream outfile ("test.txt");
7. **char** ch;
9. std::cout << "Type some text (type a dot to finish):\n";
10. **do** {
11. ch = std::cin.get();
12. outfile.put(ch);
13. } **while** (ch!='.');
15. **return** 0;
16. }

         2. 插入字符数组s的前n个字符进入流中

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ostream& write (**const** **char**\* s, streamsize n);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // Copy a file
2. #include <fstream>      // std::ifstream, std::ofstream
4. **int** main () {
5. std::ifstream infile ("test.txt",std::ifstream::binary);
6. std::ofstream outfile ("new.txt",std::ofstream::binary);
8. // get size of file
9. infile.seekg (0,infile.end);
10. **long** size = infile.tellg();
11. infile.seekg (0);
13. // allocate memory for file content
14. **char**\* buffer = **new** **char**[size];
16. // read content of infile
17. infile.read (buffer,size);
19. // write to outfile
20. outfile.write (buffer,size);
22. // release dynamically-allocated memory
23. **delete**[] buffer;
25. outfile.close();
26. infile.close();
27. **return** 0;
28. }

         3. 定位：  
         给出当前字符在输出流中的位置

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streampos tellp();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // position in output stream
2. #include <fstream>      // std::ofstream
4. **int** main () {
5. std::ofstream outfile;
6. outfile.open ("test.txt");
8. outfile.write ("This is an apple",16);
9. **long** pos = outfile.tellp();
10. outfile.seekp (pos-7);
11. outfile.write (" sam",4);
13. outfile.close();
15. **return** 0;
16. }

输出:  
         This is a sample

         4. 设置下一个在流中被插入字符的位置

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ostream& seekp (streampos pos);
2. ostream& seekp (streamoff off, ios\_base::seekdir way);

         ios\_base::seekdir定义：

|  |  |
| --- | --- |
| **member constant** | **seeking relative to** |
| beg | beginning of sequence. |
| cur | current position within sequence. |
| end | end of sequence. |

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // position in output stream
2. #include <fstream>      // std::ofstream

5. **int** main () {
7. std::ofstream outfile;
8. outfile.open ("test.txt");
10. outfile.write ("This is an apple",16);
11. **long** pos = outfile.tellp();
12. outfile.seekp (pos-7);
13. outfile.write (" sam",4);
15. outfile.close();
17. **return** 0;
18. }

**同步：**  
         1. 同步关联的输出缓冲数据进入流

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ostream& flush();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // Flushing files
2. #include <fstream>      // std::ofstream
4. **int** main () {
6. std::ofstream outfile ("test.txt");
8. **for** (**int** n=0; n<100; ++n)
9. {
10. outfile << n;
11. outfile.flush();
12. }
13. outfile.close();
15. **return** 0;
16. }

**1.5. 输入输出流（iostream）**

         iostream从ostream和istream继承，是普通输入输出流类和用于其它输入输出流的基类。iostream的接口包括了所有的ostream和istream接口。

**1.6. 输出文件流类(ofstream)**

         1. 构造函数:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ofstream();
2. **explicit** ofstream (**const** **char**\* filename, ios\_base::openmode mode = ios\_base::out);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // ofstream constructor.
2. #include <fstream>      // std::ofstream
4. **int** main () {
6. std::ofstream ofs ("test.txt", std::ofstream::out);
8. ofs << "lorem ipsum";
10. ofs.close();
12. **return** 0;
13. }

         **成员函数:**  
         1. 打开文件

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **void** open (**const** **char**\* filename,  ios\_base::openmode mode = ios\_base::out);

         mode

|  |  |  |
| --- | --- | --- |
| **member constant** | **stands for** | **access** |
| in | **in**put | File open for reading: the [*internal stream buffer*](http://www.cplusplus.com/fstream::rdbuf) supports input operations. |
| out | **out**put | File open for writing: the [*internal stream buffer*](http://www.cplusplus.com/fstream::rdbuf) supports output operations. |
| binary | **binary** | Operations are performed in binary mode rather than text. |
| ate | **at e**nd | The *output position* starts at the end of the file. |
| app | **app**end | All output operations happen at the end of the file, appending to its existing contents. |
| trunc | **trunc**ate | Any contents that existed in the file before it is open are discarded. |

         插一句：app模式下，写指针每次都会在文件结尾，ate模式则只在打开时才将写指针置于文件末尾。ate赋予了程序员修改文件内容的能力。流seekp的能力是依赖于流内部缓冲区实现的。也就说如果不赋予流读文件的能力，没有读的缓冲区，流就无法seekp到文件的任意位置。所以想要定位到文件某一位置，对文件进行修改，要如下打开文件：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. fstream(filename, ios::in|ios::out|ios::ate)。

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // ofstream::open / ofstream::close
2. #include <fstream>      // std::ofstream

5. **int** main () {
7. std::ofstream ofs;
8. ofs.open ("test.txt", std::ofstream::out | std::ofstream::app);
10. ofs << " more lorem ipsum";
12. ofs.close();
14. **return** 0;
15. }

         2. 确认文件是否打开

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **bool** is\_open();

         在函数内部调用rdbuf()->is\_open()；  
  
         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // ofstream::is\_open
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ofstream
5. **int** main () {
6. std::ofstream ofs;
7. ofs.open ("test.txt");
8. **if** (ofs.is\_open())
9. {
10. ofs << "lorem ipsum";
11. std::cout << "Output operation successfully performed\n";
12. ofs.close();
13. }
14. **else**
15. {
16. std::cout << "Error opening file";
17. }
18. **return** 0;
19. }

输出:  
         Output operation successfully performed  
  
  
         3. 关闭当前同流关联的文件

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **void** close();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // ofstream::open / ofstream::close
2. #include <fstream>      // std::ofstream
4. **int** main () {
6. std::ofstream ofs;
7. ofs.open ("test.txt", std::ofstream::out | std::ofstream::app);
9. ofs << " more lorem ipsum";
11. ofs.close();
13. **return** 0;
14. }

         4. 获取内部的文件流缓冲对象的指针

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. filebuf\* rdbuf() **const**;

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // copy a file using file stream buffers
2. #include <fstream>      // std::filebuf, std::ifstream, std::ofstream
3. #include <cstdio>       // EOF

6. **int** main () {
7. std::ifstream ifs ("test.txt");
8. std::ofstream ofs ("copy.txt");
10. std::filebuf\* inbuf  = ifs.rdbuf();
11. std::filebuf\* outbuf = ofs.rdbuf();
13. **char** c = inbuf->sbumpc();
14. **while** (c != EOF)
15. {
16. outbuf->sputc (c);
17. c = inbuf->sbumpc();
18. }
20. ofs.close();
21. ifs.close();
23. **return** 0;
24. }

**1.7. 输入文件流类(ifstream)**

         1. 构造函数:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ifstream();
2. **explicit** ifstream (**const** **char**\* filename, ios\_base::openmode mode = ios\_base::in);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // ifstream constructor.
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
7. std::ifstream ifs ("test.txt", std::ifstream::in);
9. **char** c = ifs.get();
11. **while** (ifs.good()) {
12. std::cout << c;
13. c = ifs.get();
14. }
16. ifs.close();
18. **return** 0;
19. }

**成员函数:**  
  
         1. 打开文件

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **void** open (**const** **char**\* filename,  ios\_base::openmode mode = ios\_base::in);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // print the content of a text file.
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. std::ifstream ifs;
8. ifs.open ("test.txt", std::ifstream::in);
10. **char** c = ifs.get();
12. **while** (ifs.good()) {
13. std::cout << c;
14. c = ifs.get();
15. }
17. ifs.close();
19. **return** 0;
20. }

         2. 是否文件打开

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **bool** is\_open();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // ifstream::is\_open
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. std::ifstream ifs ("test.txt");
8. **if** (ifs.is\_open()) {
9. // print file:
10. **char** c = ifs.get();
11. **while** (ifs.good()) {
12. std::cout << c;
13. c = ifs.get();
14. }
15. }
16. **else** {
17. // show message:
18. std::cout << "Error opening file";
19. }
21. **return** 0;
22. }

         3. 关闭流关联的文件

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **void** close();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // print the content of a text file.
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. std::ifstream ifs;
8. ifs.open ("test.txt");
10. **char** c = ifs.get();
12. **while** (ifs.good()) {
13. std::cout << c;
14. c = ifs.get();
15. }
17. ifs.close();
19. **return** 0;
20. }

         4. 获取内部关联的文件缓冲区指针

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. filebuf\* rdbuf() **const**;

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // read file data using associated buffer's members
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::filebuf, std::ifstream

6. **int** main () {
7. std::ifstream ifs ("test.txt", std::ifstream::binary);
9. // get pointer to associated buffer object
10. std::filebuf\* pbuf = ifs.rdbuf();
12. // get file size using buffer's members
13. std::**size\_t** size = pbuf->pubseekoff (0,ifs.end,ifs.in);
14. pbuf->pubseekpos (0,ifs.in);
16. // allocate memory to contain file data
17. **char**\* buffer=**new** **char**[size];
19. // get file data
20. pbuf->sgetn (buffer,size);
22. ifs.close();
24. // write content to stdout
25. std::cout.write (buffer,size);
27. **delete**[] buffer;
29. **return** 0;
30. }

**1.8. 输入输出文件流(fstream)**

         fstream从ofstream和ifstream继承，其接口包括了所有的ofstream和ifstream接口。

**1.9. 输出串流类(ostringstream)**

         1. 构造函数：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **explicit** ostringstream (ios\_base::openmode which = ios\_base::out);
2. **explicit** ostringstream (**const** string& str, ios\_base::openmode which = ios\_base::out);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // ostringstream constructor
2. #include <iostream>     // std::cout, std::ios
3. #include <sstream>      // std::ostringstream
5. **int** main () {
6. std::ostringstream foo;                            // out
7. std::ostringstream bar (std::ostringstream::ate);  // out|ate
9. foo.str("Test string");
10. bar.str("Test string");
12. foo << 101;
13. bar << 101;
15. std::cout << foo.str() << '\n';
16. std::cout << bar.str() << '\n';
17. **return** 0;
18. }

输出:  
         101t string  
         Test string101  
  
  
**成员函数：**  
  
         1.得到或获得字符串

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. string str() **const**;
2. **void** str (**const** string& s);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // ostringstream::rdbuf
2. #include <string>       // std::string
3. #include <iostream>     // std::cout
4. #include <sstream>      // std::ostringstream

7. **int** main () {
8. std::ostringstream oss;
9. oss << "One hundred and one: " << 101;
10. std::string s = oss.str();
11. std::cout << s << '\n';
12. **return** 0;
13. }

输出:

         One hundred and one: 101  
  
  
         2. 插入操作符

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ostream& operator<< (**bool** val);
2. ostream& operator<< (**short** val);
3. ostream& operator<< (unsigned **short** val);
4. ostream& operator<< (**int** val);
5. ostream& operator<< (unsigned **int** val);
6. ostream& operator<< (**long** val);
7. ostream& operator<< (unsigned **long** val);
8. ostream& operator<< (**float** val);
9. ostream& operator<< (**double** val);
10. ostream& operator<< (**long** **double** val);
11. ostream& operator<< (**void**\* val);
12. ostream& operator<< (streambuf\* sb );
13. ostream& operator<< (ostream& (\*pf)(ostream&));
14. ostream& operator<< (ios& (\*pf)(ios&));
15. ostream& operator<< (ios\_base& (\*pf)(ios\_base&));

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // example on insertion
2. #include <iostream>     // std::cout, std::right, std::endl
3. #include <iomanip>      // std::setw
5. **int** main () {
6. **int** val = 65;
8. std::cout << std::right;       // right-adjusted (manipulator)
9. std::cout << std::setw(10);    // set width (extended manipulator)
11. std::cout << val << std::endl; // multiple insertions
13. **return** 0;
14. }

输出:  
        65  
  
  
         3.插入字符c进入流

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ostream& put (**char** c);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // typewriter
2. #include <iostream>     // std::cin, std::cout
3. #include <fstream>      // std::ofstream
5. **int** main () {
6. std::ofstream outfile ("test.txt");
7. **char** ch;
9. std::cout << "Type some text (type a dot to finish):\n";
10. **do** {
11. ch = std::cin.get();
12. outfile.put(ch);
13. } **while** (ch!='.');
15. **return** 0;
16. }

         4.写字符串数组s的前n个字符进入流

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ostream& write (**const** **char**\* s, streamsize n);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // Copy a file
2. #include <fstream>      // std::ifstream, std::ofstream
4. **int** main () {
5. std::ifstream infile ("test.txt",std::ifstream::binary);
6. std::ofstream outfile ("new.txt",std::ofstream::binary);
8. // get size of file
9. infile.seekg (0,infile.end);
10. **long** size = infile.tellg();
11. infile.seekg (0);
13. // allocate memory for file content
14. **char**\* buffer = **new** **char**[size];
16. // read content of infile
17. infile.read (buffer,size);
19. // write to outfile
20. outfile.write (buffer,size);
22. // release dynamically-allocated memory
23. **delete**[] buffer;
25. outfile.close();
26. infile.close();
27. **return** 0;
28. }

         5.获取当前字符在流中的位置

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streampos tellp();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // position in output stream
2. #include <fstream>      // std::ofstream
4. **int** main () {
5. std::ofstream outfile;
6. outfile.open ("test.txt");
8. outfile.write ("This is an apple",16);
9. **long** pos = outfile.tellp();
10. outfile.seekp (pos-7);
11. outfile.write (" sam",4);
13. outfile.close();
15. **return** 0;
16. }

         6.设置下一个插入字符在流中的位置

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ostream& seekp (streampos pos);
2. ostream& seekp (streamoff off, ios\_base::seekdir way);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // position in output stream
2. #include <fstream>      // std::ofstream
4. **int** main () {
6. std::ofstream outfile;
7. outfile.open ("test.txt");
9. outfile.write ("This is an apple",16);
10. **long** pos = outfile.tellp();
11. outfile.seekp (pos-7);
12. outfile.write (" sam",4);
14. outfile.close();
16. **return** 0;
17. }

输出:  
         This is a sample  
  
  
         7.同步关联的流缓冲数据进入流

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. ostream& flush();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // Flushing files
2. #include <fstream>      // std::ofstream
4. **int** main () {
6. std::ofstream outfile ("test.txt");
8. **for** (**int** n=0; n<100; ++n)
9. {
10. outfile << n;
11. outfile.flush();
12. }
13. outfile.close();
15. **return** 0;
16. }

**1.10. 输入串流类(istringstream)**

         1. 构造函数:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **explicit** istringstream (ios\_base::openmode which = ios\_base::in);
2. **explicit** istringstream (**const** string& str, ios\_base::openmode which = ios\_base::in);

          例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istringstream constructors.
2. #include <iostream>     // std::cout
3. #include <sstream>      // std::istringstream
4. #include <string>       // std::string
6. **int** main () {
8. std::string stringvalues = "125 320 512 750 333";
9. std::istringstream iss (stringvalues);
11. **for** (**int** n=0; n<5; n++)
12. {
13. **int** val;
14. iss >> val;
15. std::cout << val\*2 << '\n';
16. }
18. **return** 0;
19. }

输出:  
         250  
         640  
         1024  
         1500  
         666  
  
  
**成员函数：**  
         1. 获取设置内容

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. string str() **const**;
2. **void** str (**const** string& s);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istringstream::str
2. #include <string>       // std::string
3. #include <iostream>     // std::cout
4. #include <sstream>      // std::istringstream
6. **int** main () {
7. std::istringstream iss;
8. std::string strvalues = "32 240 2 1450";
10. iss.str (strvalues);
12. **for** (**int** n=0; n<4; n++)
13. {
14. **int** val;
15. iss >> val;
16. std::cout << val << '\n';
17. }
18. std::cout << "Finished writing the numbers in: ";
19. std::cout << iss.str() << '\n';
20. **return** 0;
21. }

输出:  
         32  
         240  
         2  
         1450  
         Finished writing the numbers in: 32 240 2 1450  
  
  
         2. 返回关联的内部缓冲区指针

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. stringbuf\* rdbuf() **const**;

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istringstream::rdbuf
2. #include <string>       // std::string
3. #include <iostream>     // std::cout
4. #include <sstream>      // std::istringstream, std::stringbuf
6. **int** main () {
7. std::istringstream iss;
8. std::stringbuf \*pbuf = iss.rdbuf();
10. // using stringbuf directly:
11. pbuf->str("Example string");
13. **int** size = pbuf->in\_avail();
15. **while** (pbuf->in\_avail()>0)
16. std::cout << **static\_cast**<**char**>(pbuf->sbumpc());
18. **return** 0;
19. }

输出:  
         Example string  
  
  
         3. 重载析取操作符

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& operator>> (**bool**& val);
2. istream& operator>> (**short**& val);
3. istream& operator>> (unsigned **short**& val);
4. istream& operator>> (**int**& val);
5. istream& operator>> (unsigned **int**& val);
6. istream& operator>> (**long**& val);
7. istream& operator>> (unsigned **long**& val);
8. istream& operator>> (**float**& val);
9. istream& operator>> (**double**& val);
10. istream& operator>> (**long** **double**& val);
11. istream& operator>> (**void**\*& val);
12. istream& operator>> (streambuf\* sb );
13. istream& operator>> (istream& (\*pf)(istream&));
14. istream& operator>> (ios& (\*pf)(ios&));
15. istream& operator>> (ios\_base& (\*pf)(ios\_base&));

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // example on extraction
2. #include <iostream>     // std::cin, std::cout, std::hex
4. **int** main () {
5. **int** n;
7. std::cout << "Enter a number: ";
8. std::cin >> n;
9. std::cout << "You have entered: " << n << '\n';
11. std::cout << "Enter a hexadecimal number: ";
12. std::cin >> std::hex >> n;         // manipulator
13. std::cout << "Its decimal equivalent is: " << n << '\n';
15. **return** 0;
16. }

         4. 返回最后一次无格式输入操作所析取的字符数量

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streamsize gcount() **const**;

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // cin.gcount example
2. #include <iostream>     // std::cin, std::cout
4. **int** main () {
5. **char** str[20];
7. std::cout << "Please, enter a word: ";
8. std::cin.getline(str,20);
9. std::cout << std::cin.gcount() << " characters read: " << str << '\n';
11. **return** 0;
12. }

输出:  
         Please, enter a word: simplify  
         9 characteres read: simplify  
  
  
         5. 以无格式输入方式从输入流中析取字符

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** get();
2. istream& get (**char**& c);
3. istream& get (**char**\* s, streamsize n);
4. istream& get (**char**\* s, streamsize n, **char** delim);
5. istream& get (streambuf& sb);
6. istream& get (streambuf& sb, **char** delim);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::get example
2. #include <iostream>     // std::cin, std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. **char** str[256];
8. std::cout << "Enter the name of an existing text file: ";
9. std::cin.get (str,256);    // get c-string
11. std::ifstream is(str);     // open file
13. **while** (is.good())          // loop while extraction from file is possible
14. {
15. **char** c = is.get();       // get character from file
16. **if** (is.good())
17. std::cout << c;
18. }
20. is.close();                // close file
22. **return** 0;
23. }

         6.以无格式输入方式从输入流中析取字符，并存储于c风格的字符串中。当析取字符是分割符或者已经析取了n个字符时，动作终止。默认情况下，分割符为"\n".

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& getline (**char**\* s, streamsize n );
2. istream& getline (**char**\* s, streamsize n, **char** delim );

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::getline example
2. #include <iostream>     // std::cin, std::cout
4. **int** main () {
5. **char** name[256], title[256];
7. std::cout << "Please, enter your name: ";
8. std::cin.getline (name,256);
10. std::cout << "Please, enter your favourite movie: ";
11. std::cin.getline (title,256);
13. std::cout << name << "'s favourite movie is " << title;
15. **return** 0;
16. }

         7. 从流中提取字符输入序列(直到n个字符提取，或遇见一个指定的字符分隔字)，并丢弃。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& ignore (streamsize n = 1, **int** delim = EOF);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::ignore example
2. #include <iostream>     // std::cin, std::cout
4. **int** main () {
5. **char** first, last;
7. std::cout << "Please, enter your first name followed by your surname: ";
9. first = std::cin.get();     // get one character
10. std::cin.ignore(256,' ');   // ignore until space
12. last = std::cin.get();      // get one character
14. std::cout << "Your initials are " << first << last << '\n';
16. **return** 0;
17. }

输出:  
         Please, enter your first name followed by your surname: John Smith  
         Your initials are JS  
  
  
         8. 返回输入序列中的下一个字符，但不析取：字符将在下一次析取操作中从流中提取。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** peek();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::peek example
2. #include <iostream>     // std::cin, std::cout
3. #include <string>       // std::string
5. **int** main () {
7. std::cout << "Please, enter a number or a word: ";
8. **char** c = std::cin.peek();
10. **if** ( (c >= '0') && (c <= '9') )
11. {
12. **int** n;
13. std::cin >> n;
14. std::cout << "You entered the number: " << n << '\n';
15. }
16. **else**
17. {
18. std::string str;
19. std::getline (std::cin, str);
20. std::cout << "You entered the word: " << str << '\n';
21. }
23. **return** 0;
24. }

输出:  
         Please, enter a number or a word: foobar  
         You entered the word: foobar  
  
  
         9. 从流中析取n个字符，并把它们存储入s指向的字符数组

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& read (**char**\* s, streamsize n);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // read a file into memory
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
7. std::ifstream is ("test.txt", std::ifstream::binary);
8. **if** (is) {
9. // get length of file:
10. is.seekg (0, is.end);
11. **int** length = is.tellg();
12. is.seekg (0, is.beg);
14. **char** \* buffer = **new** **char** [length];
16. std::cout << "Reading " << length << " characters... ";
17. // read data as a block:
18. is.read (buffer,length);
20. **if** (is)
21. std::cout << "all characters read successfully.";
22. **else**
23. std::cout << "error: only " << is.gcount() << " could be read";
24. is.close();
26. // ...buffer contains the entire file...
28. **delete**[] buffer;
29. }
30. **return** 0;
31. }

输出:  
         Reading 640 characters... all characters read successfully.  
  
  
         10. 从流中析取n个字符，并把它们存储入s指向的字符数组。在内部关联的缓冲区(如果存在的话)用完时，操作返回，即使此时尚未达到文件结束字符。这个函数主要用于从某些类型的异步源中读取数据。由于析取操作会在内部缓冲区耗尽时终止，从而避免了潜在的延时。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streamsize readsome (**char**\* s, streamsize n);

         11. 把指定字符还给输入流，使得字符能够被重新析取

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& putback (**char** c);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::putback example
2. #include <iostream>     // std::cin, std::cout
3. #include <string>       // std::string
5. **int** main () {
6. std::cout << "Please, enter a number or a word: ";
7. **char** c = std::cin.get();
9. **if** ( (c >= '0') && (c <= '9') )
10. {
11. **int** n;
12. std::cin.putback (c);
13. std::cin >> n;
14. std::cout << "You entered a number: " << n << '\n';
15. }
16. **else**
17. {
18. std::string str;
19. std::cin.putback (c);
20. getline (std::cin,str);
21. std::cout << "You entered a word: " << str << '\n';
22. }
23. **return** 0;
24. }

输出:  
         Please, enter a number or a word: pocket  
         You entered a word: pocket  
  
  
         12. 把上一次析取的字符还给输入流，使得字符能够被重新析取

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& unget();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // istream::unget example
2. #include <iostream>     // std::cin, std::cout
3. #include <string>       // std::string
5. **int** main () {
6. std::cout << "Please, enter a number or a word: ";
7. **char** c = std::cin.get();
9. **if** ( (c >= '0') && (c <= '9') )
10. {
11. **int** n;
12. std::cin.unget();
13. std::cin >> n;
14. std::cout << "You entered a number: " << n << '\n';
15. }
16. **else**
17. {
18. std::string str;
19. std::cin.unget();
20. getline (std::cin,str);
21. std::cout << "You entered a word: " << str << '\n';
22. }
23. **return** 0;
24. }

输出:  
         Please, enter a number or a word: 7791  
         You entered a number: 7791  
  
  
         13. 返回当前输入流字符的位置：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streampos tellg();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // read a file into memory
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. std::ifstream is ("test.txt", std::ifstream::binary);
7. **if** (is) {
8. // get length of file:
9. is.seekg (0, is.end);
10. **int** length = is.tellg();
11. is.seekg (0, is.beg);
13. // allocate memory:
14. **char** \* buffer = **new** **char** [length];
16. // read data as a block:
17. is.read (buffer,length);
19. is.close();
21. // print content:
22. std::cout.write (buffer,length);
24. **delete**[] buffer;
25. }
27. **return** 0;
28. }

         14. 设置下一个析取字符串在输入流中的位置

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. istream& seekg (streampos pos);
2. istream& seekg (streamoff off, ios\_base::seekdir way);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // read a file into memory
2. #include <iostream>     // std::cout
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. std::ifstream is ("test.txt", std::ifstream::binary);
7. **if** (is) {
8. // get length of file:
9. is.seekg (0, is.end);
10. **int** length = is.tellg();
11. is.seekg (0, is.beg);
13. // allocate memory:
14. **char** \* buffer = **new** **char** [length];
16. // read data as a block:
17. is.read (buffer,length);
19. is.close();
21. // print content:
22. std::cout.write (buffer,length);
24. **delete**[] buffer;
25. }
27. **return** 0;
28. }

         15. 同步流相关的缓冲区buf

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** sync();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // syncing input stream
2. #include <iostream>     // std::cin, std::cout
4. **int** main () {
5. **char** first, second;
7. std::cout << "Please, enter a word: ";
8. first = std::cin.get();
9. std::cin.sync();
11. std::cout << "Please, enter another word: ";
12. second = std::cin.get();
14. std::cout << "The first word began by " << first << '\n';
15. std::cout << "The second word began by " << second << '\n';
17. **return** 0;
18. }

输出:  
         Please, enter a word: test  
         Please enter another word: text  
         The first word began by t  
         The second word began by t

**1.11. 输入输出串流类（stringstream）**

         stringstream从iostream类继承，但是接口实现上同istringstream和ostringstream一致。

**1.12. 输入输出文件流类（fstream）**

         在标准库中，fstream从iostream类继承，但是接口实现上同ifstream和ofstream。采用从ifstream和ofstream多继承的方案，个人觉得实现也可以。

**1.13. 流缓冲区(streambuf)**

         streambuf类是所有流缓冲类的基类，内部处理窄字符(narror char)。  
         缓冲区类负责关联流类的读写操作。也就说，流把所有的读写操作都委托给了缓冲区类。缓冲区类是介于流和可控的输入输出序列之间的中介。所有的流对象，不管是有缓冲的还是没有缓冲的，都关联了一个流缓冲对象。当然在流缓冲对象内部可能存在或者不存在真实的缓冲。  
  
  
        **本地化：**  
         1. 关联locale对象到缓冲区

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. locale pubimbue (**const** locale& loc);

         2. 获取当前使用的locale对象

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. locale getloc() **const**;

**缓冲管理和位置信息**  
         1. 设置缓冲区字符数组

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streambuf\* pubsetbuf (**char**\* s, streamsize n);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // set character buffer (pubsetbuf)
2. #include <fstream>      // std::fstream
4. **int** main () {
5. **char** mybuffer [512];
6. std::fstream filestr;
7. filestr.rdbuf()->pubsetbuf(mybuffer,512);
9. // operations with file stream here.
11. **return** 0;
12. }

         2. 设置内部位置指针到指定位置(相对于way的偏移)

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streampos pubseekoff (streamoff off, ios\_base::seekdir way, ios\_base::openmode which = ios\_base::in | ios\_base::out);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // get file size using pubseekoff
2. #include <iostream>     // std::cout, std::streambuf
3. #include <fstream>      // std::fstream
5. **int** main () {
6. std::fstream filestr ("test.txt");
7. **if** (filestr) {
8. std::streambuf\* pbuf = filestr.rdbuf();
9. **long** size = pbuf->pubseekoff(0,filestr.end);
10. std::cout << "The file size is " << size << " characters.\n";
11. filestr.close();
12. }
14. **return** 0;
15. }

         3. 设置内部位置指针到指定位置(绝对位置)

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streampos pubseekpos (streampos pos, ios\_base::openmode which = ios\_base::in | ios\_base::out);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // changing position with pubseekpos
2. #include <iostream>     // std::cout, std::streambuf
3. #include <fstream>      // std::fstream
5. **int** main () {
7. std::fstream filestr ("test.txt");
8. **if** (filestr) {
9. std::streambuf\* pbuf = filestr.rdbuf();
10. **long** size = pbuf->pubseekoff(0,filestr.end);  // get size
11. **if** (size>20) {
12. **char** buffer[11];
13. // change position to the 10th character
14. pbuf->pubseekpos(10);
15. // read 10 characters
16. pbuf->sgetn (buffer,10);
17. // append null character to string
18. buffer[10]=0;
19. std::cout << buffer << '\n';
20. }
21. filestr.close();
22. }
23. **return** 0;
24. }

         4. 同步流缓冲

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** pubsync();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // pubsync member
2. #include <iostream>     // std::cout, std::streambuf
3. #include <fstream>      // std::ofstream
5. **int** main () {
6. std::ofstream ostr ("test.txt");
7. **if** (ostr) {
8. std::streambuf \* pbuf = ostr.rdbuf();
10. pbuf->sputn ("First sentence\n",15);
11. pbuf->pubsync();
12. pbuf->sputn ("Second sentence\n",16);
14. ostr.close();
15. }
16. **return** 0;
17. }

         **输入函数:**  
         1. 获取可读的字符数目

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streamsize in\_avail();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // get file size using pubseekoff
2. #include <iostream>     // std::cout, std::streambuf, std::streamsize
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. std::ifstream ifs ("test.txt");
7. **if** (ifs.good()) {
8. std::streambuf\* pbuf = ifs.rdbuf();
9. **char** c; ifs >> c;
10. std::streamsize size = pbuf->in\_avail();
11. std::cout << "first character in file: " << c << '\n';
12. std::cout << size << " characters in buffer after it\n";
13. }
14. ifs.close();
16. **return** 0;
17. }

         2. 前进输入流的当前位到下一字符，并返回下一个字符

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** snextc();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // show file content - snextc() example
2. #include <iostream>     // std::cout, std::streambuf
3. #include <fstream>      // std::ifstream
4. #include <cstdio>       // EOF
6. **int** main () {
7. std::ifstream istr ("test.txt");
8. **if** (istr) {
9. std::streambuf \* pbuf = istr.rdbuf();
10. **do** {
11. **char** ch = pbuf->sgetc();
12. std::cout << ch;
13. } **while** ( pbuf->snextc() != EOF );
14. istr.close();
15. }
16. **return** 0;
17. }

         3. 返回输入流的当前位字符，并前进当前位到下一个字符

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** sbumpc();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // show file content - sbumpc() example
2. #include <iostream>     // std::cout, std::streambuf
3. #include <fstream>      // std::ifstream
4. #include <cstdio>       // EOF

7. **int** main () {
8. std::ifstream istr ("test.txt");
9. **if** (istr) {
10. std::streambuf \* pbuf = istr.rdbuf();
11. **while** ( pbuf->sgetc() != EOF )
12. {
13. **char** ch = pbuf->sbumpc();
14. std::cout << ch;
15. }
16. istr.close();
17. }
18. **return** 0;
19. }

         4. 返回输入流的当前位字符，不改变位置信息

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** sgetc();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // show file content - sgetc() example
2. #include <iostream>     // std::cout, std::streambuf
3. #include <fstream>      // std::ifstream
4. #include <cstdio>       // EOF

7. **int** main () {
8. std::ifstream istr ("test.txt");
9. **if** (istr) {
10. std::streambuf \* pbuf = istr.rdbuf();
11. **do** {
12. **char** ch = pbuf->sgetc();
13. std::cout << ch;
14. } **while** ( pbuf->snextc() != EOF );
15. istr.close();
16. }
17. **return** 0;
18. }

         5. 从指定的输入流中获取字符串，并把它们存储到s指向的字符串数组中。当n个字符已读或者输入流结束时，析取操作结束。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streamsize sgetn (**char**\* s, streamsize n);

  例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // read a file into buffer - sgetn() example
2. #include <iostream>     // std::cout, std::streambuf, std::streamsize
3. #include <fstream>      // std::ifstream
5. **int** main () {
6. **char**\* contents;
7. std::ifstream istr ("test.txt");
9. **if** (istr) {
10. std::streambuf \* pbuf = istr.rdbuf();
11. std::streamsize size = pbuf->pubseekoff(0,istr.end);
12. pbuf->pubseekoff(0,istr.beg);       // rewind
13. contents = **new** **char** [size];
14. pbuf->sgetn (contents,size);
15. istr.close();
16. std::cout.write (contents,size);
17. }
18. **return** 0;
19. }

         6. 回退输入流的位置到指定字符c的上一个位置

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** sputbackc (**char** c);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // sputbackc example
2. #include <iostream>     // std::cin, std::cout, std::streambuf, std::streamsize
3. #include <cstdio>       // EOF
5. **int** main () {
6. **char** ch;
7. std::streambuf \* pbuf = std::cin.rdbuf();
9. std::cout << "Please, enter some letters and then a number: ";
10. **do** {
11. ch = pbuf->sbumpc();
13. **if** ( (ch>='0') && (ch <='9') )
14. {
15. pbuf->sputbackc (ch);
16. **long** n;
17. std::cin >> n;
18. std::cout << "You entered number " << n << '\n';
19. **break**;
20. }
21. } **while** ( ch != EOF );
23. **return** 0;
24. }

         7. 回退输入流的位置到上一个位置，和sputbackc类似，但是不含参数c

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** sungetc();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // sputbackc example
2. #include <iostream>     // std::cin, std::cout, std::streambuf, std::streamsize
3. #include <cstdio>       // EOF
5. **int** main () {
6. **char** ch;
7. std::streambuf \* pbuf = std::cin.rdbuf();
9. std::cout << "Please, enter some letters and then a number: ";
10. **do** {
11. ch = pbuf->sbumpc();
13. **if** ( (ch>='0') && (ch <='9') )
14. {
15. pbuf->sputbackc (ch);
16. **long** n;
17. std::cin >> n;
18. std::cout << "You entered number " << n << '\n';
19. **break**;
20. }
21. } **while** ( ch != EOF );
23. **return** 0;
24. }

**输出函数:**  
         1. 存储字符于缓冲区当前位置并且移动输入指针位置

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **int** sputc (**char** c);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // typewriter - sputc() example
2. #include <iostream>     // std::cin, std::cout, std::streambuf
3. #include <fstream>      // std::ofstream
5. **int** main () {
6. **char** ch;
7. std::ofstream ostr ("test.txt");
8. **if** (ostr) {
9. std::cout << "Writing to file. Type a dot (.) to end.\n";
10. std::streambuf \* pbuf = ostr.rdbuf();
11. **do** {
12. ch = std::cin.get();
13. pbuf->sputc(ch);
14. } **while** (ch!='.');
15. ostr.close();
16. }
18. **return** 0;
19. }

         2. 存储字符串序列于缓冲区中，直到存储数目到n或者输出序列结束

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. streamsize sputn (**const** **char**\* s, streamsize n);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // sputn() example
2. #include <iostream>     // std::streambuf
3. #include <fstream>      // std::ofstream
5. **int** main () {
6. **const** **char** sentence[]= "Sample sentence";
8. std::ofstream ostr ("test.txt");
9. **if** (ostr) {
10. std::streambuf \* pbuf = ostr.rdbuf();
11. pbuf->sputn (sentence,**sizeof**(sentence)-1);
12. ostr.close();
13. }
15. **return** 0;
16. }

**1.14. 文件缓冲区流(filebuf)**

**成员函数:**  
         1. 打开文件，并关联其内容到指定的文件缓冲

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. filebuf\* open (**const** **char**\* filename,  ios\_base::openmode mode);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // filebuf::open()
2. #include <iostream>
3. #include <fstream>
5. **int** main () {
6. std::ifstream is;
7. std::filebuf \* fb = is.rdbuf();
9. fb->open ("test.txt",std::ios::out|std::ios::app);
11. // >> appending operations here <<
13. fb->close();
15. **return** 0;
16. }

         2. 判断缓冲区是否已经关联文件

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. **bool** is\_open() **const**;

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // filebuf::is\_open() example
2. #include <iostream>
3. #include <fstream>
5. **int** main () {
6. std::ifstream is;
7. std::filebuf \* fb = is.rdbuf();
8. fb->open ("test.txt",std::ios::in);
10. **if** ( fb->is\_open() )
11. std::cout << "the file is open.\n";
12. **else**
13. std::cout << "the file is not open.\n";
15. fb->close();
17. **return** 0;
18. }

         3. 关闭缓冲区关联的文件，并解除关联

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. filebuf\* close();

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // filebuf::close()
2. #include <iostream>
3. #include <fstream>
5. **int** main () {
6. std::ifstream is;
7. std::filebuf \* fb = is.rdbuf();
9. fb->open ("test.txt",std::ios::in);
11. // appending operations
13. fb->close();
15. **return** 0;
16. }

**1.15. 字符串的缓冲流类（stringbuf）**

         1. 获取设置字符串内容

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. string str() **const**;
2. **void** str (**const** string& str);

         例子:

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8698456)

1. // stringbuf example
2. #include <string>       // std::string
3. #include <iostream>     // std::cout, std::ostream, std::hex
4. #include <sstream>      // std::stringbuf
6. **int** main ()
7. {
8. std::stringbuf buffer;             // empty buffer
10. std::ostream os (&buffer);      // associate stream buffer to stream
12. // mixing output to buffer with inserting to associated stream:
13. buffer.sputn ("255 in hexadecimal: ",20);
14. os << std::hex << 255;
16. std::cout << buffer.str();
18. **return** 0;
19. }

输出:

         255 in hexadecimal: ff

**1.16 标准库流总结**

         上面啰里啰唆的讲了半天，总结如下：

         1. ios\_base封装了流的格式

         2. basic\_ios封装了流的状态信息。并定义了指向streambuf的指针，也就是说：所有的流都关联一个缓冲区。在流构造时，必须**显式的指定**一个缓冲区streambuf的指针。文件流和字符串流在构造函数里，并没有看到streambuf指针参数，并不代表不需要，在其内部封装。**所有流的输入输出操作都委托给了其内部的streambuf进行**。

         3. istream定义了输入操作，其中最重要的是重载了">>"操作符。

         4. ostream定义了输出操作，其中最重要的是重载了"<<"操作符。

         5. iostream多重继承于istream和ostream，能够同时完成读写功能。

         6. ofstream、ifstream、fstream完成文件读写工作，定义了文件open,close接口。

         7. ostringstream、istringstream、stringstream完成了对字符串的读写工作，定义了str接口。重载了"<<"，">>"操作。

         8. cerr(无缓冲标准错误)、clog(缓冲标准错误)、cout(缓冲标准输出)、cin(行缓冲)为标准库中预定义的stream对象。它们也关联有缓冲对象，其缓冲特性如下：

             cerr(无缓冲标准错误)       -----      没有缓冲，发送给它的内容立即被输出

             clog(缓冲标准错误)          -----      有缓冲，缓冲区满时输出

             cout(缓冲标准输出)          -----      标准输出

             cin(行缓冲)

附录:

在查阅C++流的相关资料时，发现有一篇文章很有意思，重点在底下的讨论。  文章链接为《[C++的流设计很糟糕](http://www.cppblog.com/converse/archive/2010/07/06/119427.html)》。

（版权所有，转载时请注明作者和出处  <http://blog.csdn.net/arau_sh/article/details/8698456>）