# C++ cout成员方法格式化输出

《[C++输入流和输出流](http://c.biancheng.net/view/7559.html)》一节中，已经针对 cout 讲解了一些常用成员方法的用法。除此之外，ostream 类中还包含一些可实现格式化输出的成员方法，这些成员方法都是从 ios 基类（以及 ios\_base  类）中继承来的，cout（以及 cerr、clog）也能调用。  
  
表 1 罗列了 ostream 类中可实现格式化输出的常用成员方法，以及它们各自的用法。

|  |  |
| --- | --- |
| 表 1 ostream 类的成员方法 | |
| **成员函数** | **说明** |
| flags(fmtfl) | 当前格式状态全部替换为 fmtfl。注意，fmtfl 可以表示一种格式，也可以表示多种格式。 |
| precision(n) | 设置输出浮点数的精度为 n。 |
| width(w) | 指定输出宽度为 w 个字符。 |
| fill(c) | 在指定输出宽度的情况下，输出的宽度不足时用字符 c 填充（默认情况是用空格填充）。 |
| setf(fmtfl, mask) | 在当前格式的基础上，追加 fmtfl 格式，并删除 mask 格式。其中，mask 参数可以省略。 |
| unsetf(mask) | 在当前格式的基础上，删除 mask 格式。 |

其中，对于表 1 中 flags() 函数的 fmtfl 参数、setf() 函数中的 fmtfl 参数和 mask 参数以及 unsetf() 函数 mask 参数，可以选择表 2 中列出的这些值。

|  |  |
| --- | --- |
| 表 2 fmtfl 和 mask 参数可选值 | |
| **标 志** | **作 用** |
| ios::boolapha | 把 true 和 false 输出为字符串 |
| ios::left | 输出数据在本域宽范围内向左对齐 |
| ios::right | 输出数据在本域宽范围内向右对齐 |
| ios::internal | 数值的符号位在域宽内左对齐，数值右对齐，中间由填充字符填充 |
| ios::dec | 设置整数的基数为 10 |
| ios::oct | 设置整数的基数为 8 |
| ios::hex | 设置整数的基数为 16 |
| ios::showbase | 强制输出整数的基数（八进制数以 0 开头，十六进制数以 0x 打头） |
| ios::showpoint | 强制输出浮点数的小点和尾数 0 |
| ios::uppercase | 在以科学记数法格式 E 和以十六进制输出字母时以大写表示 |
| ios::showpos | 对正数显示“+”号 |
| ios::scientific | 浮点数以科学记数法格式输出 |
| ios::fixed | 浮点数以定点格式（小数形式）输出 |
| ios::unitbuf | 每次输出之后刷新所有的流 |

举个例子：

1. #include <iostream>
2. **using** **namespace** std;
3. int main()
4. {
5. double a = 1.23;
6. //设定后续输出的浮点数的精度为 4
7. cout.precision(4);
8. cout <<"precision: "<< a << endl;
9. //设定后续以科学计数法的方式输出浮点数
10. cout.setf(ios::scientific);
11. cout <<"scientific："<< a << endl;
12. **return** 0;
13. }

程序执行结果为：

precision: 1.23  
scientific：1.2300e+00

注意，当 cout 采用此方式进行格式化输出时，其后不能立即输出数据，而只能像示例程序中那样，再用一个 cout 输出数据。

值得一提的是，当调用 unsetf() 或者 2 个参数的 setf() 函数时，为了提高编写代码的效率，可以给 mask 参数传递如下 3 个组合格式：

* ios::adjustfield：等价于 ios::left | ios::right | ios::internal；
* ios::basefield：等价于 ios::dec | ios::oct | ios::hex；
* ios::floatfield：等价于 ios::scientific | ios::fixed。

举个例子：

1. #include <iostream>
2. **using** **namespace** std;
3. int main()
4. {
5. double f = 123;
6. //设定后续以科学计数法表示浮点数
7. cout.setf(ios::scientific);
8. cout << f << '\n';
9. //删除之前有关浮点表示的设定
10. cout.unsetf(ios::floatfield);
11. cout << f;
12. **return** 0;
13. }

程序执行结果为：

1.230000e+02  
123

**使用流操纵算子格式化输出**

表 3 罗列了 <iomanip> 头文件中定义的一些常用的格式控制符，它们都可用于格式化输出。

|  |  |  |
| --- | --- | --- |
| 表 3 C++ 流操纵算子 | | |
| **流操纵算子** | **作  用** | |
| \*dec | 以十进制形式输出整数 | 常用 |
| hex | 以十六进制形式输出整数 |
| oct | 以八进制形式输出整数 |
| fixed | 以普通小数形式输出浮点数 |
| scientific | 以科学计数法形式输出浮点数 |
| left | 左对齐，即在宽度不足时将填充字符添加到右边 |
| \*right | 右对齐，即在宽度不足时将填充字符添加到左边 |
| setbase(b) | 设置输出整数时的进制，b=8、10 或 16 |
| setw(w) | 指定输出宽度为 w 个字符，或输入字符串时读入 w 个字符。注意，该函数所起的作用是一次性的，即只影响下一次 cout 输出。 |
| setfill(c) | 在指定输出宽度的情况下，输出的宽度不足时用字符 c 填充（默认情况是用空格填充） |
| setprecision(n) | 设置输出浮点数的精度为 n。  在使用非 fixed 且非 scientific 方式输出的情况下，n 即为有效数字最多的位数，如果有效数字位数超过 n，则小数部分四舍五人，或自动变为科学计 数法输出并保留一共 n 位有效数字。  在使用 fixed 方式和 scientific 方式输出的情况下，n 是小数点后面应保留的位数。 |
| setiosflags(mask) | 在当前格式状态下，追加 mask 格式，mask 参数可选择表 2 中的所有值。 |
| resetiosflags(mask) | 在当前格式状态下，删除 mask 格式，mask 参数可选择表 2 中的所有值。 |
| boolapha | 把 true 和 false 输出为字符串 | 不常用 |
| \*noboolalpha | 把 true 和 false 输出为 0、1 |
| showbase | 输出表示数值的进制的前缀 |
| \*noshowbase | 不输出表示数值的进制.的前缀 |
| showpoint | 总是输出小数点 |
| \*noshowpoint | 只有当小数部分存在时才显示小数点 |
| showpos | 在非负数值中显示 + |
| \*noshowpos | 在非负数值中不显示 + |
| uppercase | 十六进制数中使用 A~E。若输出前缀，则前缀输出 0X，科学计数法中输出 E |
| \*nouppercase | 十六进制数中使用 a~e。若输出前缀，则前缀输出 0x，科学计数法中输出 e。 |
| internal | 数值的符号（正负号）在指定宽度内左对齐，数值右对 齐，中间由填充字符填充。 |

注意：“流操纵算子”一栏带有星号 \* 的格式控制符，默认情况下就会使用。例如在默认情况下，整数是用十进制形式输出的，等效于使用了 dec 格式控制符。

和 cout 成员方法的用法不同，下面程序演示了表 3 中这些格式控制符的用法：

1. #include <iostream>
2. #include <iomanip>
3. **using** **namespace** std;
4. int main()
5. {
6. //以十六进制输出整数
7. cout << hex << 16 << endl;
8. //删除之前设定的进制格式，以默认的 10 进制输出整数
9. cout << resetiosflags(ios::basefield)<< 16 << endl;
10. double a = 123;
11. //以科学计数法的方式输出浮点数
12. cout << scientific << a << endl;
13. //删除之前设定的科学计数法的方法
14. cout << resetiosflags(ios::scientific) << a << endl;
15. **return** 0;
16. }

程序执行结果为：

10  
16  
1.230000e+02  
123

注意，如果两个相互矛盾的标志同时被设置，如先设置 setiosflags(ios::fixed)，然后又设置 setiosflags(ios::scientific)，那么结果可能就是两个标志都不起作用。因此，在设置了某标志，又要设置其他与之矛盾的标志时，就应该用 resetiosflags 清除原先的标志。