####C++ Trick
>1.C++ 11
```
std::to_string() 要 g++ -std=c++11
g++ -std=c++11 -w file.cpp -o file  使用c++11标准，并且忽略所有警告
```
>2.有人也许以为i++或者++i的效率比拆开之后要高，这只是一种错觉。这些代码经过基本的编译器优化之后，生成的机器代码是完全没有区别的。自增减表达式只有在两种情况下才可以安全的使用。一种是在for循环的update部分，比如for(int i

>3.有人可能会说，全都打上花括号，只有一句话也打上，多碍眼啊？然而经过实行这种编码规范几年之后，我并没有发现这种写法更加碍眼，反而由于花括号的存在，使得代码界限明确，让我的眼睛负担更小了。

>4.避免使用continue和break。循环语句（for，while）里面出现return是没问题的，然而如果你使用了continue或者break，就会让循环的逻辑和终止条件变得复杂，难以确保正确。
出现continue或者break的原因，往往是对循环的逻辑没有想清楚。如果你考虑周全了，应该是几乎不需要continue或者break的。如果你的循环里出现了continue或者break，你就应该考虑改写这个循环。改写循环的办法有多种：
```
如果出现了continue，你往往只需要把continue的条件反向，就可以消除continue。
如果出现了break，你往往可以把break的条件，合并到循环头部的终止条件里，从而去掉break。
有时候你可以把break替换成return，从而去掉break。
如果以上都失败了，你也许可以把循环里面复杂的部分提取出来，做成函数调用，之后continue或者break就可以去掉了。
```
>5.如果你把异常catch了，忽略掉，那么你就不知道foo其实失败了。这就像开车时看到路口写着“前方施工，道路关闭”，还继续往前开。这当然迟早会出问题，因为你根本不知道自己在干什么。catch异常的时候，你不应该使用Exception这么宽泛的类

>6.不要把null放进“容器数据结构”里面。所谓容器（collection），是指一些对象以某种方式集合在一起，所以null不应该被放进Array，List，Set等结构，不应该出现在Map的key或者value里面。把null放进容器里面，是一些莫名其妙错误的来源。因为
解决方案是：如果你真要表示“没有”，那你就干脆不要把它放进去（Array，List，Set没有元素，Map根本没那个entry），或者你可以指定一个特殊的，真正合法的对象，用来表示“没有”。

>7

```C++

char num[128];

cin >> setw(127) >> ... 限定输入字符数

```

>8

但是inline本身还是有另外一个意义：
一个可执行文件的cpp文件中一个函数只能被定义一次。如果你把函数定义在一个.h文件中并让两个cpp包含就会造成这个函数分别在两个cpp中被定义产生错误。但是inline函数是允许在多个cpp中多次定义的，就解决了这个问题。