**30 多年的软件经验，总结出 10 个编写出更好代码的技巧**

那么，何以敲出一手好代码？

好代码可以定义为易读、易懂、易调试、易改，最重要的还要少缺陷。显然，要想敲出一手好代码，是要花不少时间，但这在长久来看是有意义的，因为你可以花更少时间及精力去维护和复用你的代码。

事实上，我们可以将好代码等同于可复用的代码，这也是下面提到的重要原则之一。代码可能只是完成了编程工作中短期目标的特定功能，但如果没人（包括你自己）愿意复用你的代码，这代码在某种程度上可以说是不足且有缺陷的。要么太复杂、要么太具体、要么在不同情况下极有可能崩掉，或者其他程序员可能不相信你的代码。

下面无论你的经验水平如何，如果你始终如一地将下面的妙招应用到你的代码中（包括你的实验或者原型），那么一手好代码随手可得。

### 1、 遵循单一责任原则

函数在程序员的库中是单一最重要的抽象形式。可以被复用的机会越多，你要写的代码就越少，这些代码就越可靠。遵循[单一责任原则](https://en.wikipedia.org/wiki/Single_responsibility_principle)的小小函数更有可能被重新使用。

### 2、最小化共享状态

应该将函数之间的隐式共享状态最小化，无论它是文件作用域变量还是对象的成员字段，这有利于显式地将所需的值作为参数。当明确函数实现所需结果时，代码变得容易理解和重用。

对此可以得出一个结论，你应该优先选择静态无状态变量而不是对象的成员变量。

### 3、本地化副作用

理想的副作用（例如打印到控制台、记录、改变全局状态、文件系统操作等）应该放置在单独的模块中，而不是分散在整个代码中。功能上的副作用往往违反了单一的责任原则。

### 4、优先选择不可变的对象

如果一个对象的状态在其构造函数中设置一次，并且不再次更改，则调试变得容易得多，因为一旦构造正确就保持有效。这是降低软件项目复杂性的最简单方法之一。

### 5、多用接口少用类

接受接口的函数（或C++中的模板参数或概念）比在类上操作的函数可重用性更强。

### 6、对模块应用良好的原则

将软件项目分解成更小的模块（例如库和应用程序），以实现模块化重用。模块的一些关键原则是：

1. 最小化依赖关系
2. 每个项目都应该有一个单一明确的功能
3. 不要重复

你应该努力让你的项目保持小巧和明确。

### 7.避免继承

在面向对象编程中，继承，特别是虚拟函数在可重用性方面往往是一个死穴。我很少能成功地使用能覆盖类的库。

### 8.同设计和开发一样进行测试

我并不是测试驱动开发的铁杆拥护者，但在你开始编写测试代码时，编写测试自然遵循了许多指导方针。它也有助于早点将错误暴露出来。避免编写无用的测试，良好的编码意味着更高级的测试（例如，单元测试中的集成测试或功能测试）在显示缺陷方面更有效。

### 9.优先选择而不是手写标准库

我无法告诉你需要多久才能看到一个 std :: vector 或 std :: string 更好的版本，但它几乎总是浪费时间和精力。除了一个显而易见的事实，那就是你正在把 bug 引入一个新的地方。（见技巧10）其他程序员不太可能重用您的代码，而不是那些被广泛理解、支持和测试的代码。

### 10.避免写新代码

最重要的一点是，每位程序员应遵循：“**The best code is the code that isn’t written**”（最好的代码是不用被复写的代码）。你的代码越多，缺陷就越多，找到并修复 bug 就越困难。

在编写一行代码之前先问问自己，有没有一个工具，函数或库已经做了你所需要的功能？你真的需要自己去实现这个功能，而不是调用另一个已经存在的功能吗？   
总结   
编程就好比是一种艺术形式或者一项运动，你只有通过不断地练习，不断地向他人学习，才能不断地提高代码的质量，这些都将有利于你成为更加高效的程序员。