C++代码规范

作者：fengsharp

QQ：493894652

Email：csdnfor@126.com

2016/5/23

目 录

1. 代码质量的理解 1

2. 零Bug 1

3. 代码结构清晰 1

4. 代码简洁易懂 1

4.1. 命名 1

# 代码质量的理解

代码质量分为很多个方面，我认为比较重要的、需要达到的目标有：

1. 零Bug
2. 结构清晰稳定，容易扩展
3. 代码简洁易懂

# 零Bug

很难写出没有Bug的代码，造成Bug的原因，我认为有几方面常见：

1.写的代码和原意不符。比如【==】判断写出了【=】赋值，宽高参数传入后，方法内部赋值却分别赋给了高和宽的成员变量等。此种情况是最不应该发生的，但是会发生，有时候90%的情况都不会出现问题，但是10%的情况肯定是大问题。

2.逻辑结构不清晰。当功能逻辑复杂的时候，写的代码如果逻辑混乱，类、方法之间的耦合度较高，状态变量特别多，则一旦功能需求有稍微的改动，则很容易因为一个小地方的改动而引入新的Bug，或者修改了一种情况的Bug，但是却产生了新的Bug，代码也不容易被别人读懂。

代码有无Bug，依赖于测试是不靠谱的，黑盒测试并不能说明代码就是真的没有Bug了。所以我认为有一种情况可以做：

1. 写代码之前，分析好都有哪些场景（测试用例）
2. 代码写完之后，遍历每一种测试用例，想办法写测试代码，跟踪代码中的各种状态，确保和设计中的逻辑一致无误。这样至少第一种情况是可以大概率规避的。

# 代码结构清晰

有几个原则可以遵循：

1.一个类的职责尽量保证只有一个，多余1个的复杂逻辑代码一定要分拆出去。这样做的主要原因在于一个职责，只需要维持一种状态成员变量，各种接口也只针对此一种情况，而多余一个的职责，就多了一份状态的维护，一旦两种状态之间有关联，则一个接口需要同时考虑多种情况，很容易出Bug。

2.类的对外接口一定要最小化原则，内部重要核心数据不能直接对外公开，要有限制的提供接口进行访问。

3.判断一个类和另一个类之间的耦合关系复杂度如何。这个我还没有想好，正在看资料。

# 代码简洁易懂

## 命名

1. 类：首字母大写，反映“What”
2. 方法：首字母小写，反映“How”。参数不能过多。代码尽量保持仅仅一个职责原则，代码行数不易太多，尽量保证在一屏以内。
3. 成员变量命名规范

|  |  |  |
| --- | --- | --- |
| 基础类型 | 前缀类型 | Demo |
| bool | b | m\_bChanging  bConverted |
| int | i | m\_iNum iLen |
| unsigned int | n | m\_nItemCode nItemCode |
| float | f | m\_fPositonX   fWidth |
| char | c | char\* m\_cName;   const char\* cMsg |
| string | str | m\_strName    const string& strMsg |
| vector 数组 | arr | m\_arrNames |
| list | lst | m\_lstNames |
| map | dic | m\_dicNameCodes |