## 使用卫语句取代嵌套表达式

函数中的条件逻辑使人难以看懂正常的执行路径，使用卫语句表现所有复杂的情况，将函数的单入口单出口变为单入口多出口。

动机：条件表达式通常有两种表现形式:

第一种：所有分支都属于正常行为

第二种：条件表达式提供的答案中只有一种正常行为，其他都是不常见的情况

这两类条件表达式有不同的用途，如果两条分支都是正常行为，就应该使用形如if…else…的表达式，如果某种条件极为罕见，就应该单独检查条件，并在该条件为真时立刻从函数中返回，这样的检查常常被称为“卫语句”。

使用卫语句取代嵌套表达式精髓是给某个特殊分支以特别的重视，它告诉阅读者：这种情况很罕见，如果它真的发生了，请做一些必要的整理工作然后退出。

**卫语句就是把复杂的条件表达式拆分成多个条件表达式，比如一个很复杂的表达式，嵌套了好几层的if - then-else语句，转换为多个if语句，实现它的逻辑，这多条的if语句就是卫语句.**

**有时候条件式可能出现在嵌套n次才能真正执行，其他分支只是简单报错返回的情况，对于这种情况，应该单独检查报错返回的分支，当条件为真时立即返回，这样的单独检查就是卫语句（guard clauses).卫语句可以把我们的视线从异常处理中解放出来，集中精力到正常处理的代码中。**

1. void func(void)
2. {
3. if(IsWorkDay())
4. {
5. printf("Error,is work day");
6. }
7. else
8. {
9. if(IsWorkTime())
10. {
11. printf("Error ,is work time");
12. }
13. else
14. {
15. rest();
16. }
17. }
18. }

使用卫语句替换后

1. void func()
2. {
3. if(IsWorkDay())
4. {
5. printf("Error,is work day");
6. return;
7. }
8. if(IsWorkTime())
9. {
10. printf("Error,is work time");
11. return ;
12. }
14. rest();
16. }

总结：当if…else 表达式嵌套层数过多时（大于三层），考虑将多层嵌套拆分层多条if语句，多条if语句称为“卫语句”，拆分的原则是将特殊的分支拆分出来单独做处理，或者将可以直接结束函数的分支拿出来做返回，增加代码的可读性。

## Map初始化、扩容

map 在第一次put的时候初始化，扩容时大小为2的n次幂，默认为16装载因子默认为0.75，第一次扩容的条件是元素个数达到16\*0.75=12在Jdk 1.7和Jdk 1.8中，HashMap初始化这个容量的时机不同。jdk1.8中，在调用HashMap的构造函数定义HashMap的时候，就会进行容量的设定。而在Jdk 1.7中，要等到第一次put操作时才进行这一操作。在jdk1.8中，如果我们传入的初始化容量为1，实际上设置的结果也为1java开发手册建议 map初始化容量设置为 初始化容量要设置成expectedSize/0.75 + 1的话，可以有效的减少冲突也可以减小误差expected：需要存储的元素个数在日常开发中，可以使用Map<String, String> map = Maps.newHashMapWithExpectedSize(10);来创建一个HashMap，计算的过程guava会帮我们完成。