**参考：**<http://blog.csdn.net/jw903/article/details/45506777>

**1.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清正常的执行途径。使用卫语句表现所有特殊情况。

动机：条件表达式通常有2种表现形式。第一：所有分支都属于正常行为。**第二：条件表达式提供的答案中只有一种是正常行为，其他都是不常见的情况。**

       这2类条件表达式有不同的用途。**如果2条分支都是正常行为，就应该使用形如if…..else…..的条件表达式；如果某个条件极其罕见，就应该单独检查该条件，并在该条件为真时立刻从函数中返回。这样的单独检查常常被称为“卫语句”**。

       Replace Nested Conditional with Guard Clauses （以卫语句取代嵌套条件表达式）的精髓是：给某个分支以特别的重视。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。

       “每个函数只能有一个入口和一个出口”的观念，根深蒂固于某些程序员的脑海里。现今的编程语言都会强制保证每个函数只有一个入口，至于“单一出口”规则，其实不是那么有用。保持代码清晰才是最关键的：如果单一出口能使这个函数更清晰易读，那么就使用单一出口；否则就不必这么做。

做法：1、对于每个检查，放进一个卫语句。卫语句要不就从函数返回，要不就抛出一个异常。

       2、每次将条件检查替换成卫语句后，编译并**[测试](http://lib.csdn.net/base/softwaretest" \o "软件测试知识库" \t "_blank)**。如果所有卫语句都导致相同的结果，请使用 Consolidate Conditional Expression （合并条件表达式）。

**2.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.**

**3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。**

**例如下列代码：**

**[html]** [view plain](http://blog.csdn.net/jw903/article/details/45506777) [copy](http://blog.csdn.net/jw903/article/details/45506777)

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. }

使用卫语句替换以后

**[html]** [view plain](http://blog.csdn.net/jw903/article/details/45506777) [copy](http://blog.csdn.net/jw903/article/details/45506777)

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. }

参考：http://blog.csdn.net/u013378469/article/details/23341635

根据《重构》第9章中可知，1.条件表达式通常有两种表现形式。第一种是所有分支都属于正常行为；第二种是条件表达式提供的答案只有一种是正常行为，其他都不是常见的情况。所以这两种表达式有不同用途，对于第一种代码可以用if-then-else的条件表达式；对于第二种某个条件极其罕见，就应该单独检查该条件，这种单独检查就被称为“卫语句”（guard clauses）。书中举例说明：

**[java]** [view plain](http://blog.csdn.net/u013378469/article/details/23341635) [copy](http://blog.csdn.net/u013378469/article/details/23341635)

1. **public** **double** getPayAmount() {
2. **if** (isDead()) **return** deadPayAmount();
3. **if** (isSeparated()) **return** separatedPayAmount();
4. **if** (isRetired()) **return** retiredPayAmount();
5. **return** normalPayAmount();
7. }

任何人都看的懂以上代码（只要理解英文单词即可），因为条件判断与结果产生一种映射关系，大脑里不用堆砌任何代码栈，傻瓜式看代码就行。