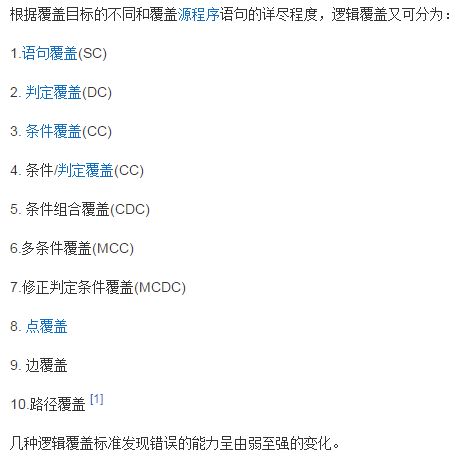
# 逻辑测试

逻辑覆盖是以程序内部的逻辑结构为基础的设计测试用例的技术。它属白盒测试。它是一系列测试过程的总称，这组测试过程逐渐进行越来越完整的通路测试。



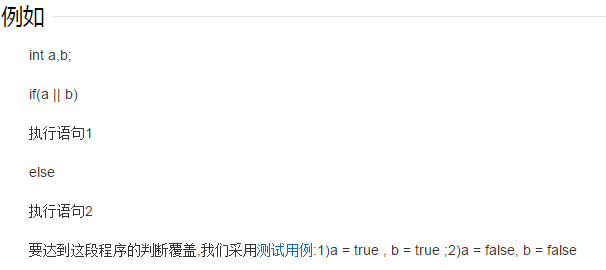
## 判定覆盖

所谓判定覆盖，就是设计若干个测试用例，运行被测程序，使得程序中每个判定条件的取真分支和取假分支至少评价一次。又称为分支覆盖。

使设计的测试用例保证程序中每个判断的每个取值分支（t or f）至少经历一次

【优点】 判定覆盖具有比语句覆盖更强的测试能力，而且具有和语句覆盖一样的简单性，无需细分每个判定就可以得到测试用例。

【缺点】 往往大部分的判定语句是由多个逻辑条件组合而成（如，判定语句中包含AND、OR、CASE），若仅仅判断其整个最终结果，而忽略每个条件的取值情况，必然会遗漏部分测试路径。



也就是说，是覆盖一个整体的逻辑判断（比如if(A&B)）取true或者false就可以了，而不关心A和B具体取什么值。

## 条件覆盖

条件覆盖是指选择足够的测试用例，使得运行这些测试用例后，要使每个判断中每个条件的可能取值至少满足一次，但未必能覆盖全部分支。（比如 A or B，可以取A true B false 和A false B true，最后的整体结果都是true）

对于判断/分支覆盖而言，即使一个布尔表达式含有多个逻辑表达式我们也只需要测试每个布尔表达式的值分别为真和假两种情况就可以了。条件覆盖要检查每个符合谓词的子表达式值为真和假两种情况，要独立衡量每个子表达式的结果，以确保每个子表达式的值为真和假两种情况都被测试到。

## 判定/条件覆盖

判定-条件覆盖就是设计足够的测试用例，得使判断中每个条件的所有可能取值至少执行一次，同时每个判断本身所有可能结果也至少执行一次。

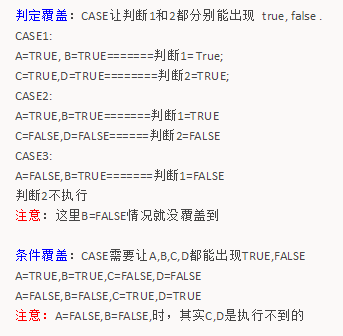
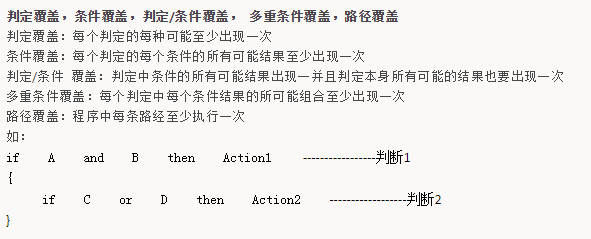
缺点：忽略了条件的组合情况。

## ~~组合覆盖~~

~~在白盒测试法中，选择足够的测试用例，使得每个判定中条件的各种可能组合都至少出现一次。显然，满足“条件组合覆盖”的测试用例是一定满足“判定覆盖”、“条件覆盖”和“判定/条件覆盖”的。~~

## 多重条件覆盖

每个判定中每个条件结果的所可能组合至少出现一次



## 修正判定条件覆盖（MC/DC）

<http://www.kailesoft.com/help/TestBox/MCDC.htm>

MC/DC是DO-178B Level A认证标准中规定的，欧美民用航空器强制要求遵守该标准。  
  
    MC/DC定义如下：  
    Condition —— a Boolean expression containing no Boolean operators：  
    Decision —— a Boolean expression composed of conditions and zero or more Boolean operators：  
    Modified Condition/Decision Coverage —— every point of entry and exit in the program has been invoked at least once，every condition in the program has taken all possible outcomes at least once，and each condition in a decision has been shown to independently affect a decision S outcome by varying just that condition while holding fixed all other possible conditions.  
  
    条件表示不含有布尔操作符号的布尔表达式；  
    判定表示由条件和零或者很多布尔操作符号所组成的一个布尔表达式；  
    修正条件判定覆盖要求在一个程序中每一种输入输出至少得出现一次，在程序中的每一个条件必须产生所有可能的输出结果至少一次，并且每一个判定中的每一个条件必须能够独立影响一个判定的输出，即在其他条件不变的前提下仅改变这个条件的值，而使判定结果改变。  
  
    上面的定义有点不好理解，解释一下：  
    MC/DC首先要求实现条件覆盖、判定覆盖，在此基础上，对于每一个条件C，要求存在符合以下条件的两次计算：  
    1）条件C所在判定内的所有条件，除条件C外，其他条件的取值完全相同；  
    2）条件C的取值相反；  
    3）判定的计算结果相反。  
  
    核心意思是每个条件都要独立影响判定结果。为什么说“两次计算”，而不是“两个用例”呢？当循环中有判定时，一个用例下同一判定可能被计算多次，每次的条件值和判定值也可能不同，因此，一个用例就可能完成循环中判定的MC/DC。  
  
    MC/DC是条件组合覆盖的子集。条件组合覆盖要求覆盖判定中所有条件取值的所有可能组合，需要大量的测试用例，实用性较差。MC/DC具有条件组合覆盖的优势，同时大幅减少用例数。满足MC/DC的用例数下界为条件数+1，上界为条件数的两倍，例如，判定中有三个条件，条件组合覆盖需要8个用例，而MC/DC需要的用例数为4至6个。如果判定中条件很多，用例数的差别将非常大，例如，判定中有10个条件，条件组合覆盖需要1024个用例，而MC/DC只需要11至20个用例。  
  
    下面是MC/DC的示例：

代码：  
    int func(BOOL A, BOOL B, BOOL C)  
    {  
        if(A && (B || C))  
            return 1;  
        return 0;  
    }  
  
    用例：

