白盒测试

# 测试技术

设计测试方案是测试阶段的关键技术问题。所谓测试方案包括：

* + 1. 具体的测试目的
    2. 应该输入的数据
    3. 预期的结果

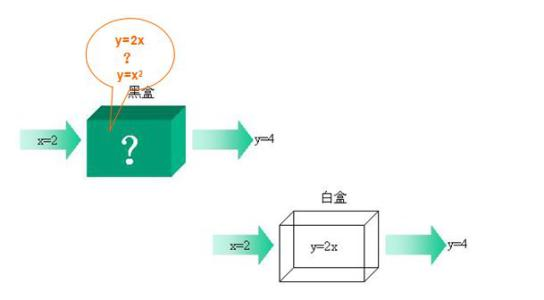
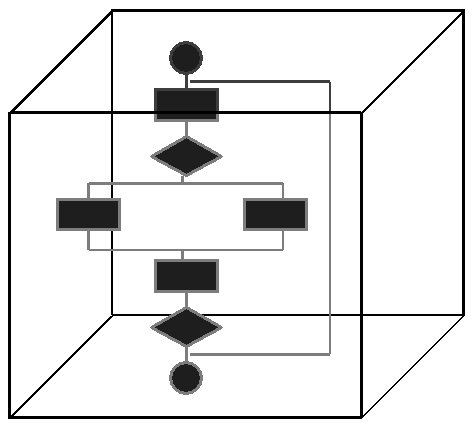
通常把**测试数据**和**输出结果**称为**测试用例**。其中最困难的问题是**设计测试用的输入数据。**

不同的数据发现程序错误的能力差别很大，为了提高效率降低成本，应该选用高效的测试数据。

**设计测试方案**的**基本目标**是，确定一组最可能发现某个错误或某类错误的测试数据。这些技术各有优缺点，没有最好的技术，也没有可以互相代替的技术；同一种技术在不同场合效果可能差别很大，因此，通常需要联合使用多种设计测试数据的技术。

# 白盒简介

白盒测试:又称结构测试、透明盒测试、逻辑驱动测试或基于代码的测试。白盒测试是一种测试数据设计方法，盒子指的是被测试的软件。白盒指的是软件是可视的，你清楚软件内部的东西以及里面是如何运作的。相对应的，黑盒指的是软件是不可视的，你不清楚软件的东西如何运作。

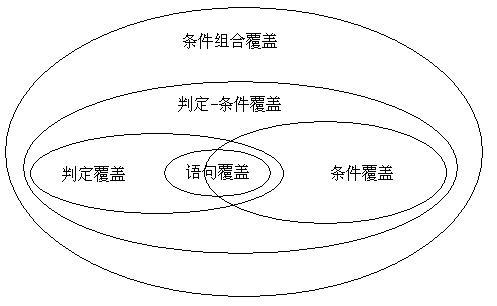


白盒测试法中最主要的有**逻辑覆盖**和**控制结构测试**。

# 逻辑覆盖

## 逻辑覆盖概述

|  |  |
| --- | --- |
| 语句覆盖（点覆盖） | 语句覆盖每条语句至少执行一次。 |
| 判定覆盖（边覆盖） | 判定覆盖每个判定的每个分支至少执行一次。 |
| 条件覆盖 | 条件覆盖每个判定的每个条件应取到各种可能的值。 |
| 判定/条件覆盖 | 判定/条件覆盖同时满足判定覆盖条件覆盖。 |
| 条件组合覆盖 | 条件组合覆盖每个判定中各条件的每一种组合至少出现一次。 |
| 路径覆盖 | 路径覆盖使程序中每一条可能的路径至少执行一次。 |



六种覆盖标准发现错误的能力呈由弱到强的变化。

**要求：**

1.保证一个模块中的所有独立路径至少被使用一次。

2.对所有逻辑值均需测试 true 和 false。

3.在上下边界及可操作范围内运行所有循环。

4.检查内部数据结构以确保其有效性。

## 语句覆盖（点覆盖）

语句覆盖：语句覆盖就是设计若干个测试用例，使被测试程序中的每条可执行语句至少执行一次。

## 判定覆盖（边覆盖）

判定覆盖（也称为分支覆盖）：设计若干个测试用例，使被测程序中每个判定的取真分支和取假分支至少执行一次。

## 条件覆盖

条件覆盖：设计足够多的测试用例，使被测程序中每个判定的每个条件的每个可能取值至少执行一次。

## 判定/条件覆盖

判定-条件覆盖：设计足够多的测试用例，使被测程序中每个判定的每个条件的所有可能取值至少执行一次，并且每个判定的取真分支和取假分支也至少执行一次。

## 条件组合覆盖

条件组合测试：设计足够多的测试用例，使被测程序中每个判定的所有可能的条件取值组合至少执行一次。

## 路径覆盖

路径覆盖：运行所测程序，要覆盖程序中所有可能的路径。

# 控制结构测试

## 基本路径测试

## 条件测试

## 循环测试

# 白盒测试优缺点

**优点**

1.迫使测试人员去仔细思考软件的实现

2.可以检测代码中的每条分支和路径

3.揭示隐藏在代码中的错误

4.对代码的测试比较彻底

5.最优化

**缺点**

1.昂贵

2.无法检测代码中遗漏的路径和数据敏感性错误

3.不验证规格的正确性

**局限**

但即使每条路径都测试了仍然可能有错误。可能出现的情况如下：

1.穷举路径测试决不能查出程序违反了设计规范，即程序本身是个错误的程序。

2.穷举路径测试不可能查出程序中因遗漏路径而出错。

3.穷举路径测试可能发现不了一些与数据相关的错误。