**广 州 商 学 院**

**实验报告（第 4 次）**

实验名称 条件组合法和程序插装法 实验时间 2025年4月30日

同组同学 无 小组分工 无

**一、实验目的**

1**.**熟悉并掌握白盒测试中的条件组合覆盖法和程序插桩法。

2.掌握条件组合覆盖法和程序插桩法测试用例的设计，编写测试用例。

3.能对实际案例进行分析，灵活运用两种测试方法进行测试用例设计。

**二、实验仪器设备或材料**

1. 计算机
2. 设计测试用例表
3. **实验原理**

1. 画出流程图

2.计算基本路径的数量，找到基本路径

3.根据基本路径设计测试用例

4.制作基本路径法的测试用例表

**四、实验内容与步骤**

**条件组合覆盖法：实验题一：登录验证的条件组合覆盖测试**

题目描述：给定以下登录验证函数，使用条件组合覆盖法设计测试用例。

public String login(String username, String password, boolean isVerified) {

if (username == null || username.isEmpty()) {

return "用户名不能为空";

}

if (password == null || password.isEmpty()) {

return "密码不能为空";

}

if (!isVerified) {

return "账号未验证";

}

if (username.equals("admin") && password.equals("Admin@123")) {

return "管理员登录成功";

}

if (username.equals("user") && password.equals("User@123")) {

return "普通用户登录成功";

}

return "用户名或密码错误";

}

**条件组合覆盖法：****实验题二：信用卡审批的条件组合覆盖测试**

题目描述：分析以下信用卡审批函数并设计条件组合覆盖测试用例。

def credit\_approval(age, income, has\_employment, credit\_score):

# 先检查拒绝条件

if age < 18:

return "年龄不足"

if not has\_employment:

return "无稳定工作"

if credit\_score < 600:

return "信用评分不足"

# 然后检查批准条件，从最高级别开始检查

if income >= 60000 and credit\_score >= 800:

return "批准白金卡"

if age > 25 and income >= 40000 and credit\_score >= 750:

return "批准金卡"

if age >= 18 and age <= 25 and income >= 20000:

return "批准标准卡"

# 最后是默认批准

return "批准普通卡"

**程序插桩法实验题设计**

**实验题三：简单函数执行路径跟踪**

**题目描述**：对以下计算阶乘的函数进行插桩，要求跟踪函数执行路径和变量变化情况。

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

**实验题四：复杂条件分支覆盖率分析**

**题目描述**：对以下信用卡审批函数进行插桩，统计各条件分支的执行情况。

def credit\_approval(age, income, employed, credit\_score):

if age < 18:

return "年龄不足"

if not employed:

return "无稳定工作"

if credit\_score < 600:

return "信用评分不足"

if income >= 60000 and credit\_score >= 800:

return "批准白金卡"

if age > 25 and income >= 40000 and credit\_score >= 750:

return "批准金卡"

if age >= 18 and income >= 20000:

return "批准标准卡"

return "批准普通卡"

**条件组合覆盖法实验步骤：**

1．识别条件

2.确定条件组合

3.设计测试用例

**程序插桩法实验步骤：**

1.分析代码，设计插桩

2.插桩代码运行

3.设计测试用例

**五、实验结果与分析**

**（1）实验题一：登录验证的条件组合覆盖测试**

1. 识别条件：

C1: username == null || username.isEmpty()

C2: password == null || password.isEmpty()

C3: !isVerified

C4: username.equals("admin") && password.equals("Admin@123")

C5: username.equals("user") && password.equals("User@123")

2、确定条件组合：

需要覆盖所有条件的真/假组合，但由于存在互斥条件，实际有效组合如下：

* + 1. C1为真
    2. C1为假且C2为真
    3. C1为假且C2为假且C3为真
    4. C1为假且C2为假且C3为假且C4为真
    5. C1为假且C2为假且C3为假且C4为假且C5为真
    6. C1为假且C2为假且C3为假且C4为假且C5为假
       1. 设计测试用例：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例 | username | password | isVerified | 预期输出 | 覆盖条件组合 |
| 1 | null | "pass" | true | "用户名不能为空" | C1为真 |
| 2 | "admin" | null | true | "密码不能为空" | C1为假且C2为真 |
| 3 | "admin" | "Admin@123" | false | "账号未验证" | C1为假且C2为假且C3为真 |
| 4 | "admin" | "Admin@123" | true | "管理员登录成功" | C1为假且C2为假且C3为假且C4为真 |
| 5 | "user" | "User@123" | true | "普通用户登录成功" | C1为假且C2为假且C3为假且C4为假且C5为真 |
| 6 | "guest" | "Guest@123" | true | "用户名或密码错误" | C1为假且C2为假且C3为假且C4为假且C5为假 |

**（2）实验题二：信用卡审批的条件组合覆盖测试**

1、识别条件：

C1: age < 18

C2: not has\_employment

C3: credit\_score < 600

C4: income >= 60000 and credit\_score >= 800

C5: age > 25 and income >= 40000 and credit\_score >= 750

C6: age >= 18 and age <= 25 and income >= 20000

2、确定条件组合：

1. C1为真

2.C1为假且C2为真

3.C1为假且C2为假且C3为真

4.C1为假且C2为假且C3为假且C4为真

5.C1为假且C2为假且C3为假且C4为假且C5为真

6.C1为假且C2为假且C3为假且C4为假且C5为假且C6为真

7.C1为假且C2为假且C3为假且C4为假且C5为假且C6为假

3、设计测试用例：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试**  **用例** | **age** | **Not Has-employment** | **credit\_**  **score** | **income** | **预期输出** | **覆盖条件组合** |
| 1 | 16 |  |  |  | "年龄不足" | C1为真 |
| 2 | 20 | True |  |  | "无稳定工作" | C1为假且C2为真 |
| 3 | 22 | False | 550 |  | "信用评分不足" | C1为假且C2为假且C3为真 |
| 4 | 22 | False | 810 | 61000 | "批准白金卡" | C1为假且C2为假且C3为假且C4为真 |
| 5 | 30 | False | 780 | 45000 | "批准金卡" | C1为假且C2为假且C3为假且C4为假且C5为真 |
| 6 | 20 | False | 650 | 30000 | "批准标准卡" | C1为假且C2为假且C3为假且C4为假且C5为假且C6为真 |
| 7 | 28 | False | 650 | 10000 | "批准普通卡" | C1为假且C2为假且C3为假且C4为假且C5为假且C6为假 |

**（3）实验题三：简单函数执行路径跟踪**

1、插桩设计：

在函数入口处插入打印语句，显示输入参数

在每个返回点前插入打印语句，显示返回值

在递归调用前后插入打印语句，显示递归深度和中间结果

2、插桩后的代码：

def factorial(n, depth=0):

print(f"{' '\*depth}-> factorial({n})") # 入口插桩

if n == 0:

print(f"{' '\*depth}<- return 1") # 出口插桩

return 1

else:

print(f"{' '\*depth} computing {n} \* factorial({n-1})") # 递归前插桩

result = n \* factorial(n-1, depth+1)

print(f"{' '\*depth}<- return {result}") # 递归后插桩

return result

# 测试调用

print("计算5的阶乘:")

factorial(5)

3.设计测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 用例编号 | 输入参数(n) | 预期返回值 | 预期打印输出（关键片段） | 说明 |
| C1 | 0 | 1 | -> factorial(0) <- return 1 | 验证递归终止条件 |
| C2 | 1 | 1 | -> factorial(1) computing 1 \* factorial(0) -> factorial(0) <- return 1 <- return 1 | 验证单层递归计算逻辑 |
| C3 | 5 | 120 | 缩进深度逐层增加： -> factorial(5) computing 5 \* factorial(4) -> factorial(4) ... -> factorial(0) <- return 1 逐层返回乘积结果 | 验证多层递归和缩进格式 |

**（4）实验题四：复合逻辑条件的基本路径测试**

1、插桩设计：

为每个条件判断添加计数器

记录每个条件分支的执行情况

在函数结束时输出覆盖率报告

2、插桩后的代码：

coverage = {

'age<18': 0,

'not employed': 0,

'credit\_score<600': 0,

'白金卡条件': 0,

'金卡条件': 0,

'标准卡条件': 0,

'普通卡': 0

}

def credit\_approval(age, income, employed, credit\_score):

if age < 18:

coverage['age<18'] += 1

return "年龄不足"

if not employed:

coverage['not employed'] += 1

return "无稳定工作"

if credit\_score < 600:

coverage['credit\_score<600'] += 1

return "信用评分不足"

if income >= 60000 and credit\_score >= 800:

coverage['白金卡条件'] += 1

return "批准白金卡"

if age > 25 and income >= 40000 and credit\_score >= 750:

coverage['金卡条件'] += 1

return "批准金卡"

if age >= 18 and income >= 20000:

coverage['标准卡条件'] += 1

return "批准标准卡"

coverage['普通卡'] += 1

return "批准普通卡"

def print\_coverage():

print("\n=== 分支覆盖率报告 ===")

total = sum(coverage.values())

for condition, count in coverage.items():

print(f"{condition:20}: {count}次 ({count/total\*100:.1f}%)")

# 测试用例集

test\_cases = [

(16, 0, True, 700), # 年龄不足

(20, 25000, False, 700), # 无稳定工作

(22, 25000, True, 550), # 信用评分不足

(30, 65000, True, 820), # 白金卡

(30, 45000, True, 780), # 金卡

(22, 25000, True, 700), # 标准卡

(20, 15000, True, 700) # 普通卡

]

# 执行测试

for i, (age, income, employed, score) in enumerate(test\_cases, 1):

print(f"\n测试用例{i}: age={age}, income={income}, employed={employed}, score={score}")

result = credit\_approval(age, income, employed, score)

print(f"审批结果: {result}")

# 打印覆盖率报告

print\_coverage()

3.设计测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用例编号 | 年龄(age) | 收入(income) | 就业状态(employed) | 信用评分(score) | 预期审批结果 | 说明 |
| C1 | 16 | 0 | True | 700 | 年龄不足 | 验证年龄下限规则1 |
| C2 | 20 | 25000 | False | 700 | 无稳定工作 | 验证就业状态强制要求3 |
| C3 | 22 | 25000 | True | 550 | 信用评分不足 | 验证评分最低阈值1 |
| C4 | 30 | 65000 | True | 820 | 批准白金卡 | 验证高收入+高评分组合3 |
| C5 | 30 | 45000 | True | 780 | 批准金卡 | 验证中等收入+良好评分组合1 |
| C6 | 22 | 25000 | True | 700 | 批准标准卡 | 验证基础参数达标场景3 |
| C7 | 20 | 15000 | True | 700 | 批准普通卡 | 验证低收入合规场景1 |

**六、结论与体会**

通过本次实验，我们掌握了白盒测试中的条件组合覆盖法和程序插桩法，并学会了设计相应的测试用例进行测试。这些方法可以帮助我们更全面地测试系统，发现潜在的问题并提高软件质量。

**七、教师评语**